# Automi two-way e complessità descrizionale

Giovanni Pighizzini

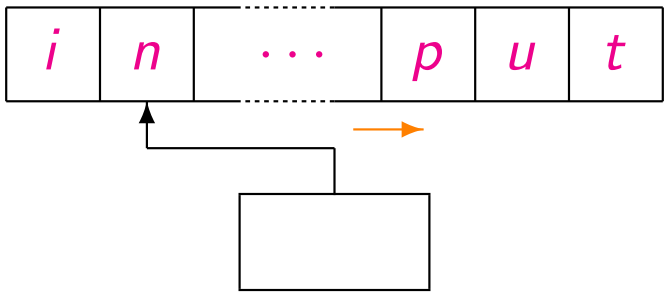
Dipartimento di Informatica
Università degli Studi di Milano

Palermo, 5 marzo 2014

UNIVERSITÀ DEGLI STUDI
DI MILANO

# Alberto Bertoni
(1946–2014)

# Automi a stati finiti




*Versione one-way:*

a ogni passo la testina viene spostata a destra di una posizione

- 1DFA: transizioni *deterministiche*
- 1NFA: transizioni *nondeterministiche*

# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

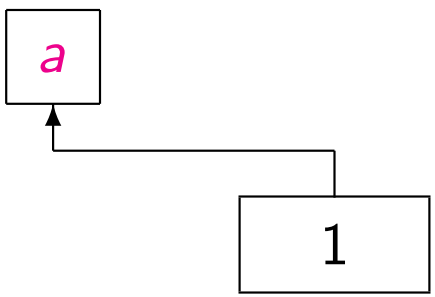
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$



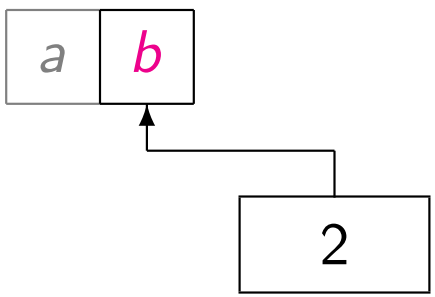
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$



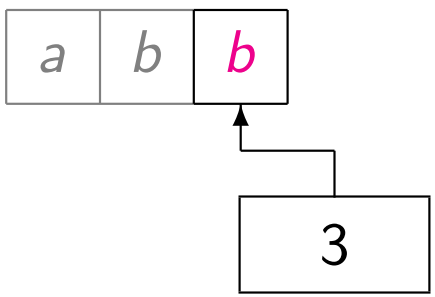
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$



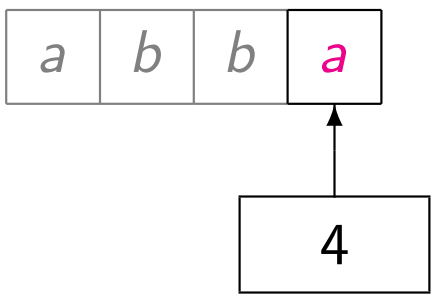
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$



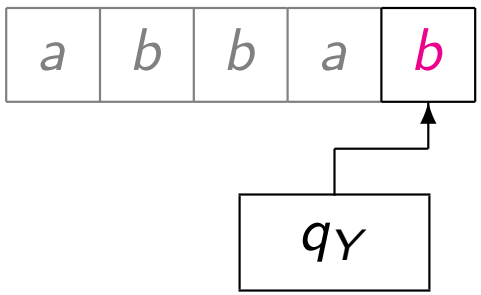
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$



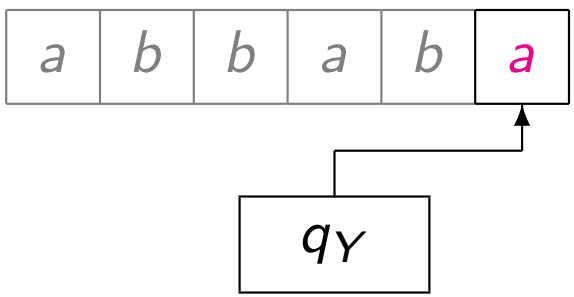
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$



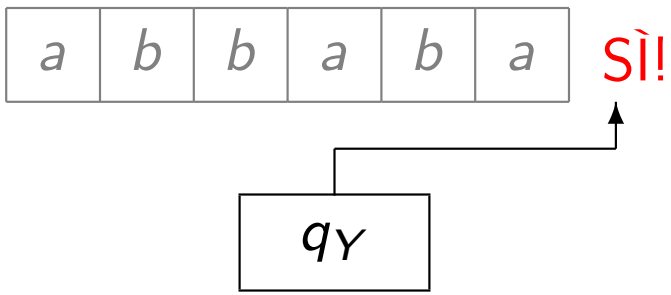
# Esempio 0

$\Sigma = \{a, b\}$, $n > 0$:

$$H_n = (a + b)^{n-1} a (a + b)^*$$

Controlla l'$n$-esimo simbolo da sinistra!

Es. $n = 4$




1DFA: $n + 2$ stati

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

*Come posso individuarlo?*

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$L_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

*Come posso individuarlo?*

Usa il nondeterminismo!

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$L_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

*Come posso individuarlo?*

Usa il nondeterminismo!

*Guess* Quando l'automa legge una $a$ può "scommettere" che sia l'$n$-esimo simbolo da destra

*Verify* Nei passi successivi verifica la scommessa

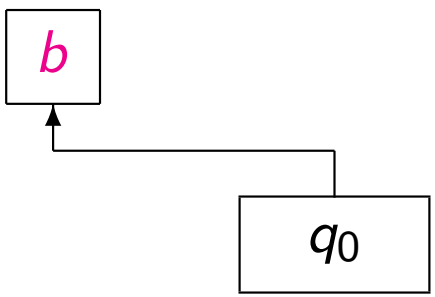
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



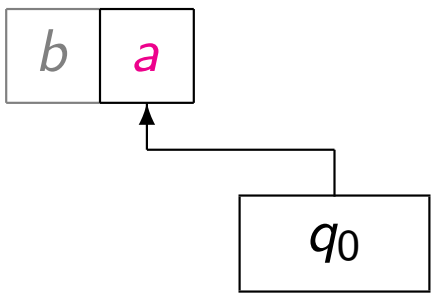
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



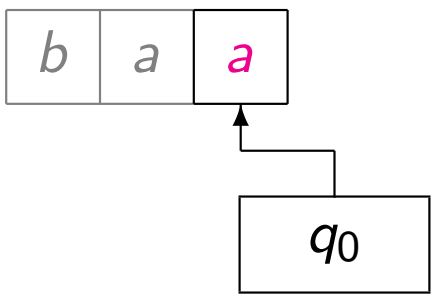
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$L_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$




*guess*

$4^\circ$ simbolo da destra

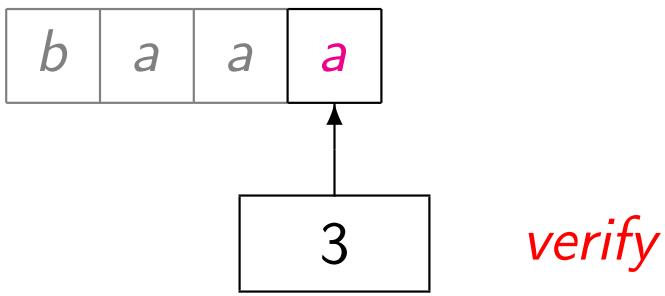
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



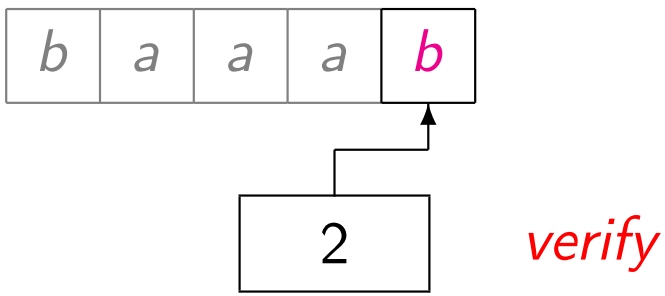
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



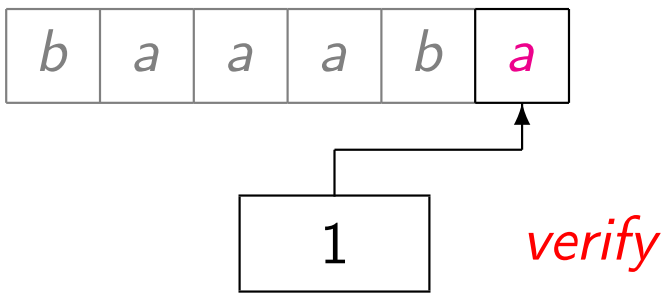
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



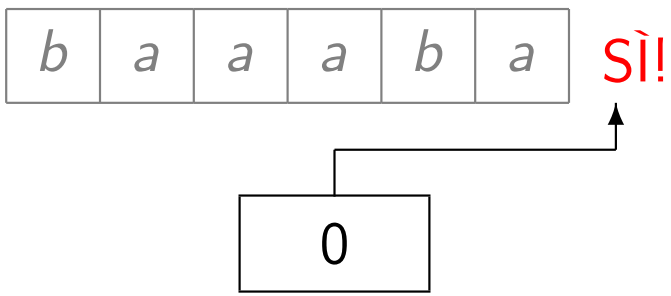
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



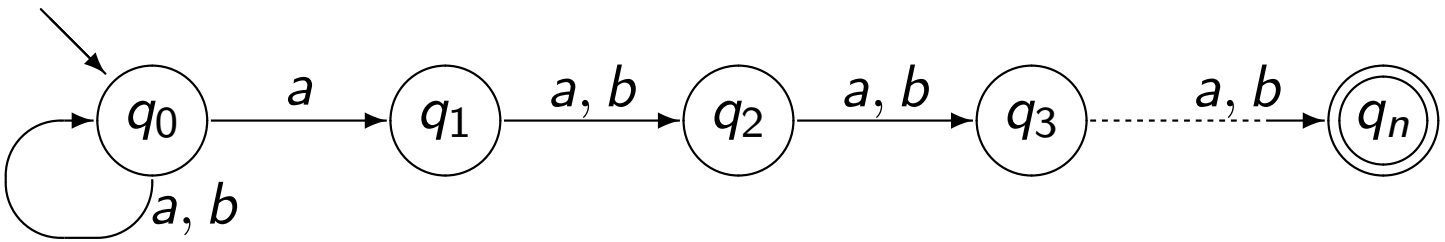

1NFA: $n + 1$ stati

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

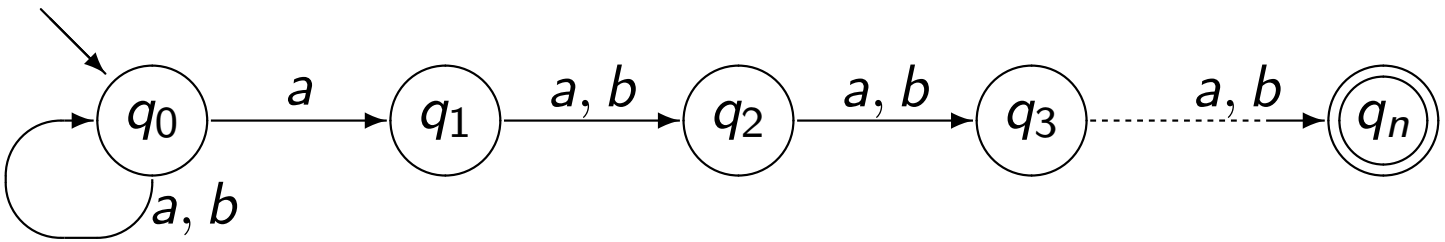


# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!




Bene!

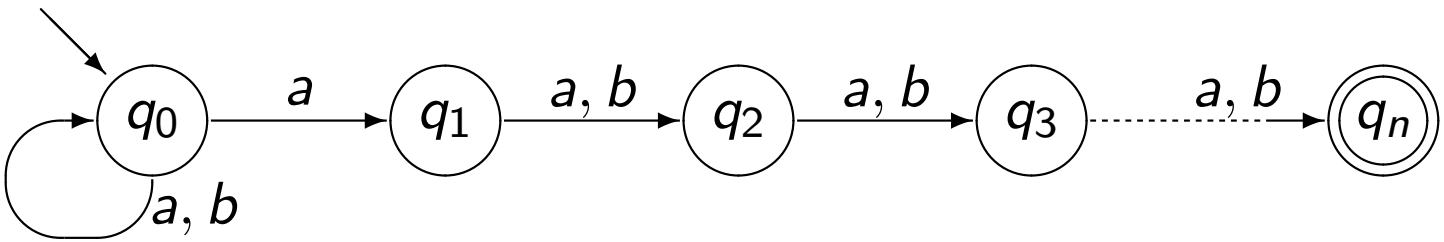
...ma mi serve un automa *deterministico*...

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!




Bene!

...ma mi serve un automa *deterministico*...

Ad ogni passo ricorda gli ultimi $n$ simboli letti!

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$

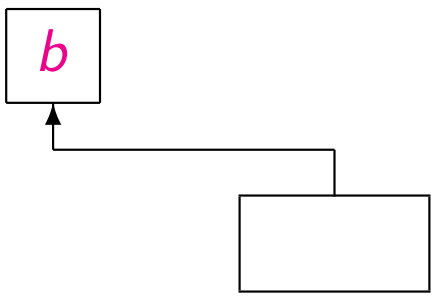
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$L_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



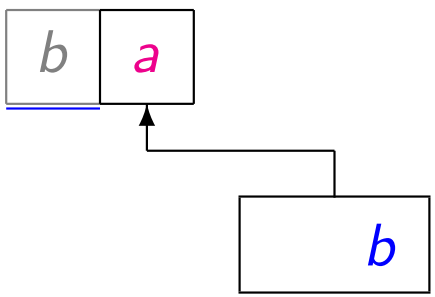
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



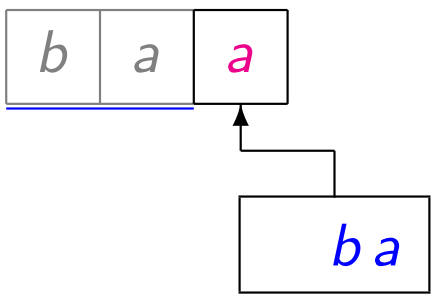
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$L_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



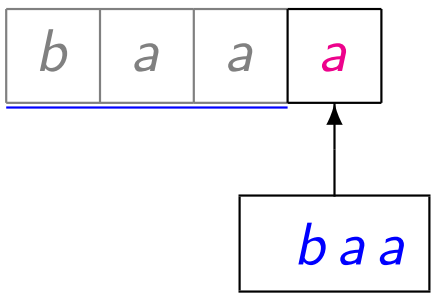
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



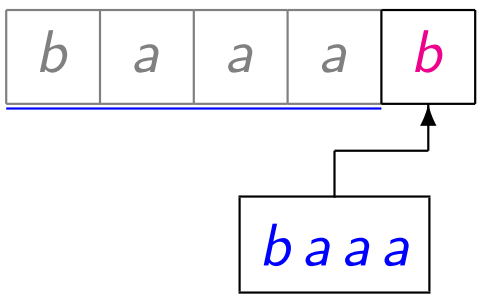
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



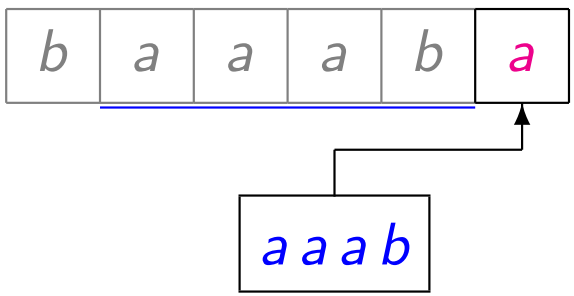
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



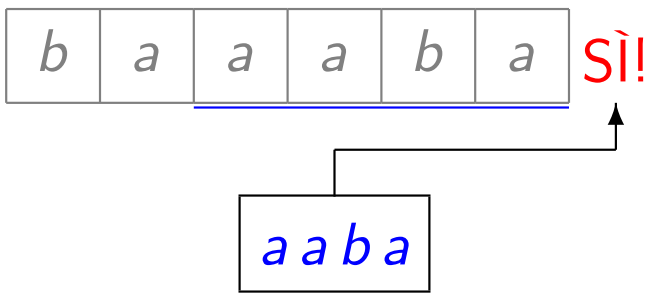
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$



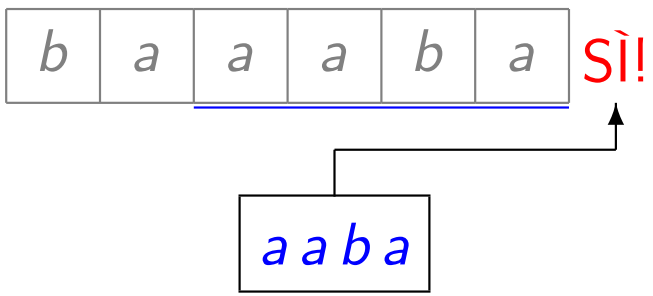
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$




1DFA: $2^n$ stati

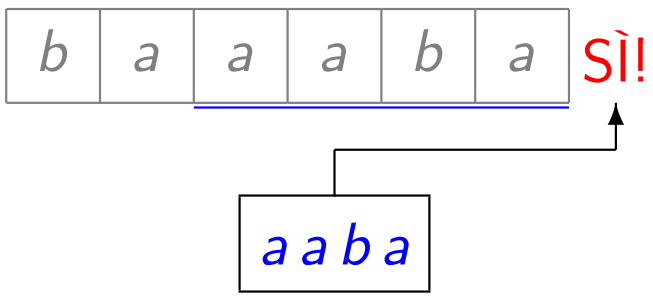
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$




1DFA: $2^n$ stati

...ma mi serve un automa deterministico più piccolo...

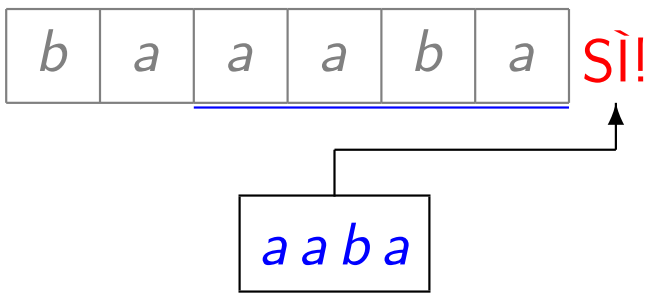
# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

Es. $n = 4$




1DFA: $2^n$ stati

...ma mi serve un automa deterministico più piccolo...

Il più piccolo è questo!

Tuttavia...

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

# Esempio 1

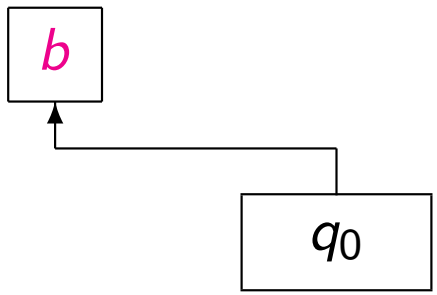
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

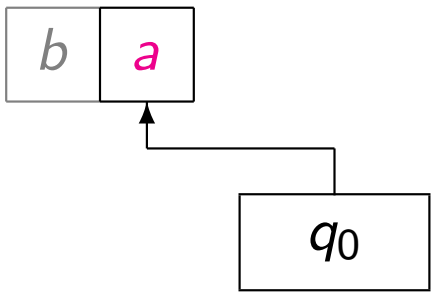
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

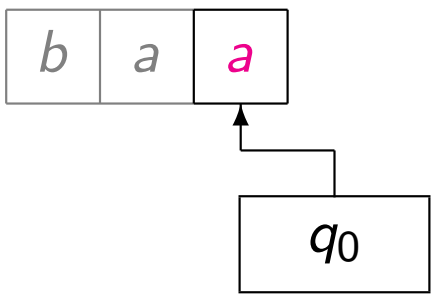
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

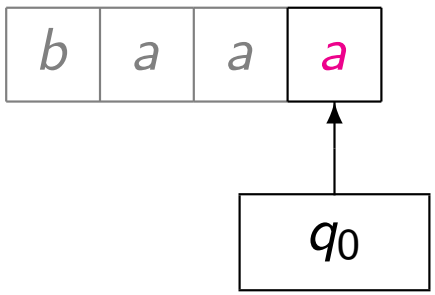
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

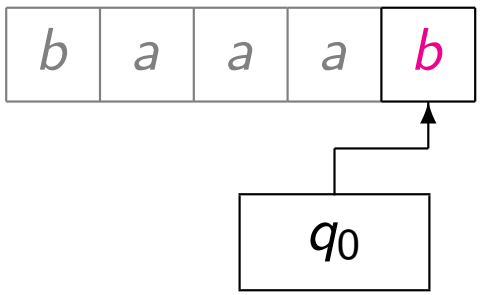
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

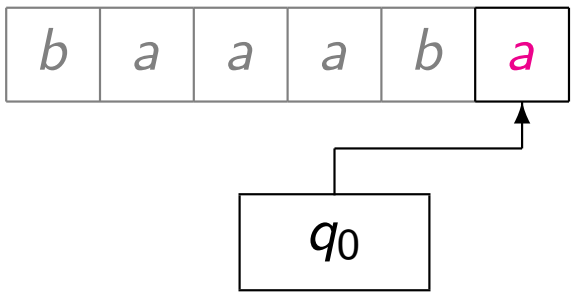
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

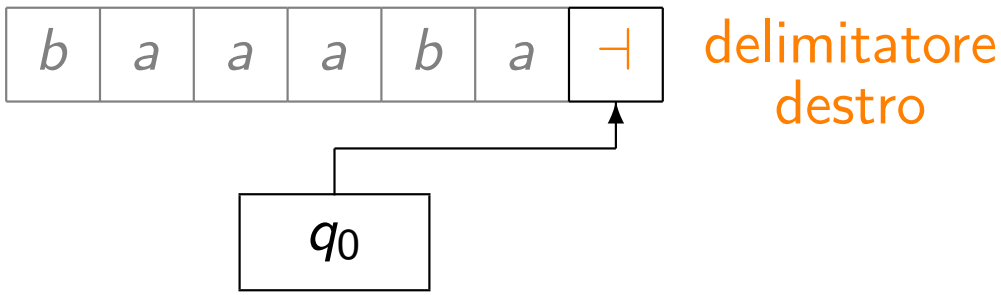
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

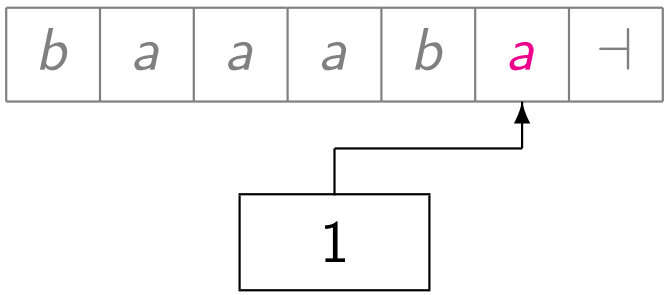
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

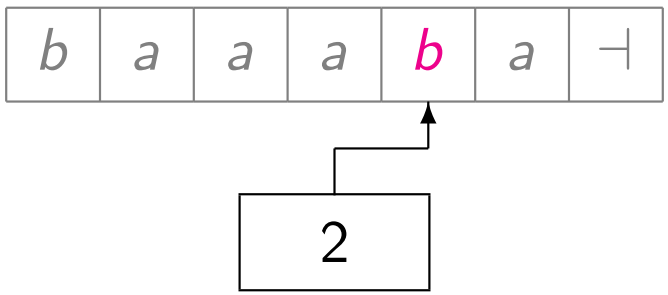
$\Sigma = \{a, b\}$, $n > 0$:

$$L_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

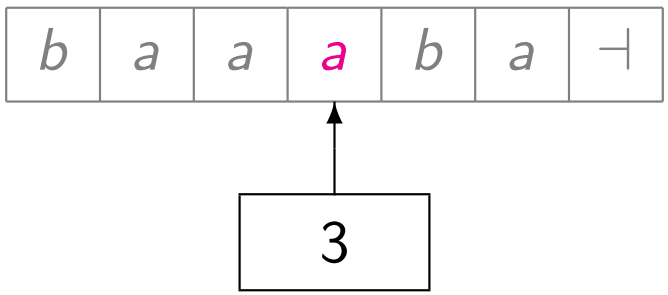
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$



# Esempio 1

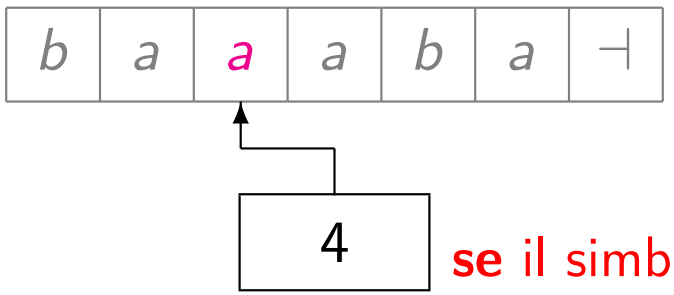
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$




**se** il simbolo è $a$ **allora accetta**
**altrimenti rifiuta**

# Esempio 1

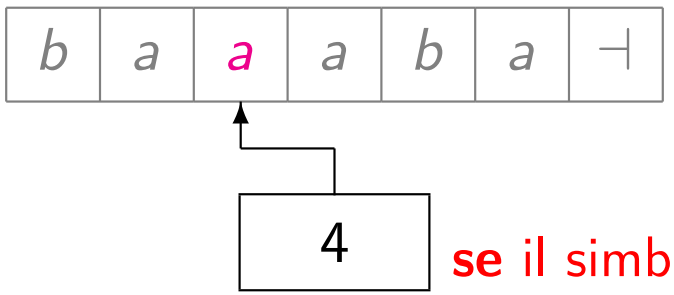
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$




SÌ!

**se** il simbolo è $a$ **allora accetta**
**altrimenti rifiuta**

# Esempio 1

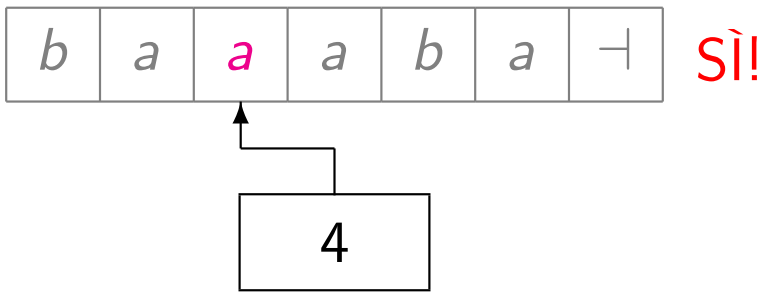
$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

...se si può muovere la testina all'indietro...

Es. $n = 4$




SÌ!

Automa deterministico *two-way* (2DFA): $n+_{\ldots}$ stati

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

Controlla l'$n$-esimo simbolo da destra!

$l_n$ è accettato da

- 1NFA e 2DFA con circa
  lo stesso numero di stati $n+\ldots$
- ogni 1DFA è esponenzialmente più grande ($\geq 2^n$ stati)

# Esempio 1

$\Sigma = \{a, b\}$, $n > 0$:

$$l_n = (a + b)^* a (a + b)^{n-1}$$

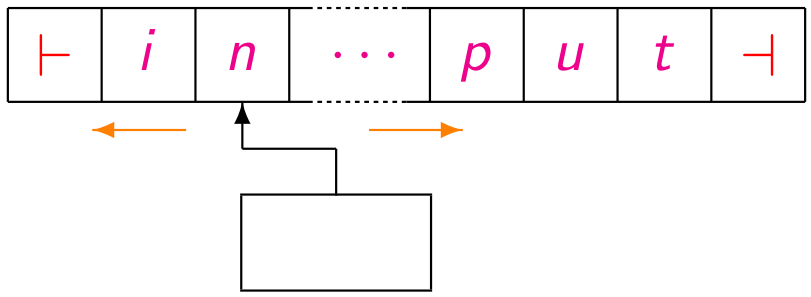
Controlla l'$n$-esimo simbolo da destra!

$l_n$ è accettato da

- 1NFA e 2DFA con circa
  lo stesso numero di stati $n+\ldots$
- ogni 1DFA è esponenzialmente più grande ($\geq 2^n$ stati)

*In questo esempio,*
il nondeterminismo può essere eliminato tornando indietro
e mantenendo approssimativamente lo stesso numero di stati

# Automi two-way: qualche dettaglio tecnico




- Due *delimitatori* dell'input $\vdash$ and $\dashv$
- Mosse
  - a *sinistra*
  - a *destra*
  - *stazionarie*
- Configurazione iniziale
- Configurazione accettante
- *Possibilità di computazioni infinite*
- Versioni *deterministica* (2DFA) e *nondeterministica* (2NFA)

# 1DFA, 1NFA, 2DFA, 2NFA

Che potenza hanno questi modelli?

# 1DFA, 1NFA, 2DFA, 2NFA

Che potenza hanno questi modelli?

Caratterizzano tutti la classe dei *linguaggi regolari*,

# 1DFA, 1NFA, 2DFA, 2NFA

Che potenza hanno questi modelli?

Caratterizzano tutti la classe dei *linguaggi regolari*, tuttavia...

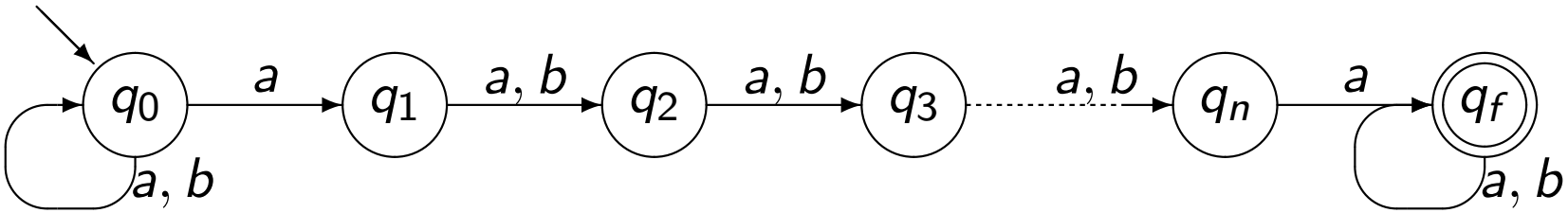
...alcuni sono più succinti

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



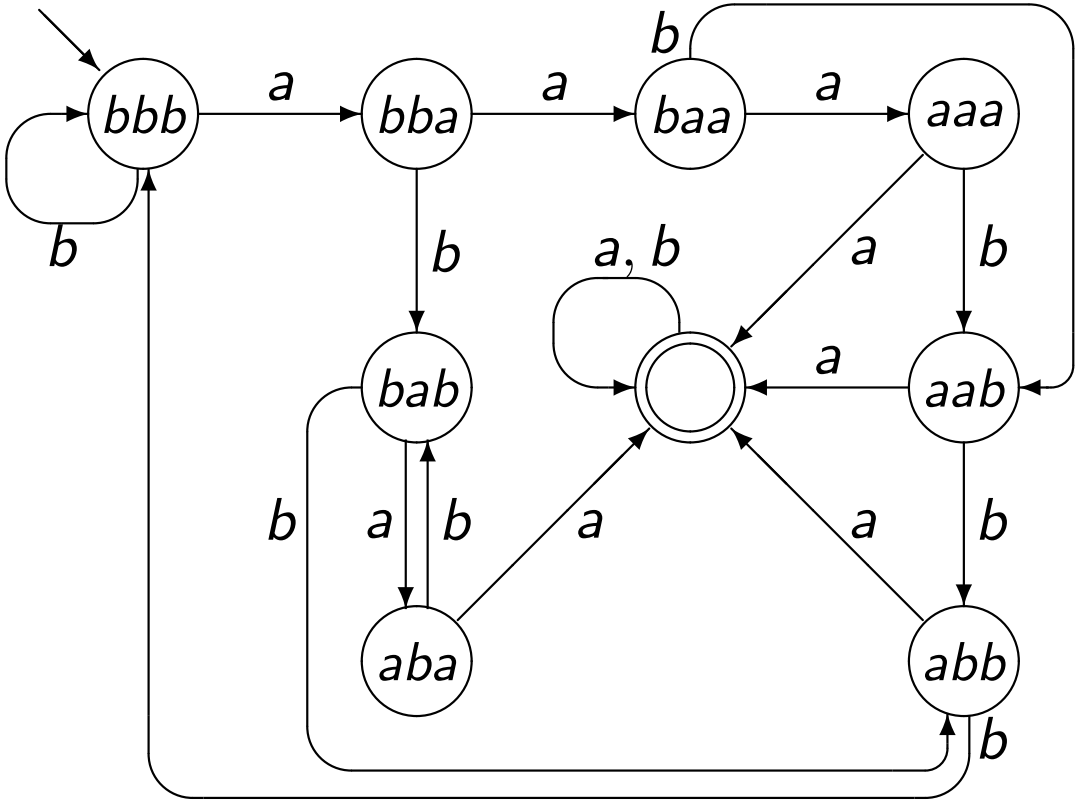

1NFA: $n + 2$ stati

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




1NFA: $n + 2$ stati

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 3$

1DFA minimo: $2^n + 1$ stati

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

2DFA ?

Anche leggendo da destra,
sembra necessario ricordare una "finestra" di $n$ simboli

Tecnica differente!

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

2DFA ?

Anche leggendo da destra,
sembra necessario ricordare una "finestra" di $n$ simboli

Tecnica differente!

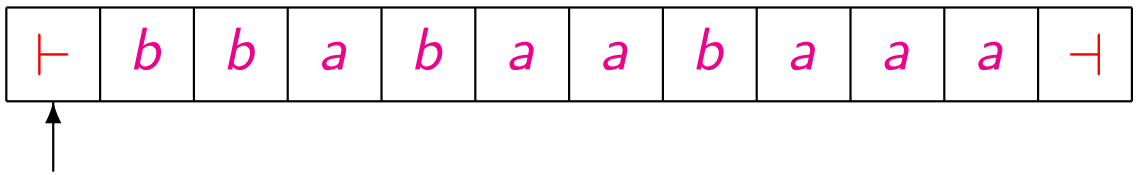
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

2DFA ?

Anche leggendo da destra,
sembra necessario ricordare una "finestra" di $n$ simboli

Tecnica differente!

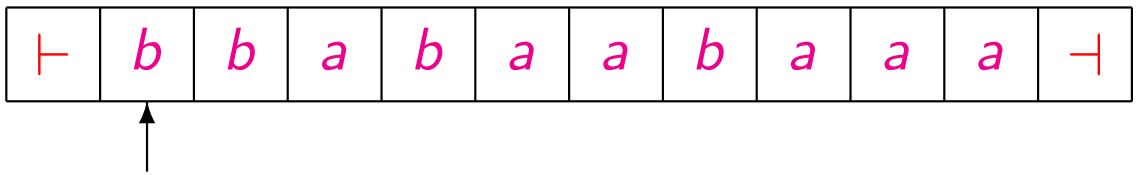
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

**finché** non trovi una $a$ muovi a destra

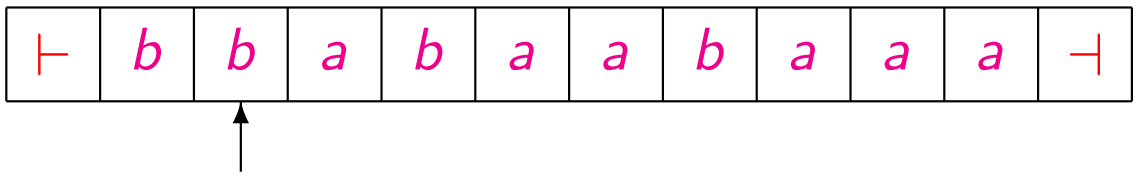
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

**finché** non trovi una $a$ muovi a destra

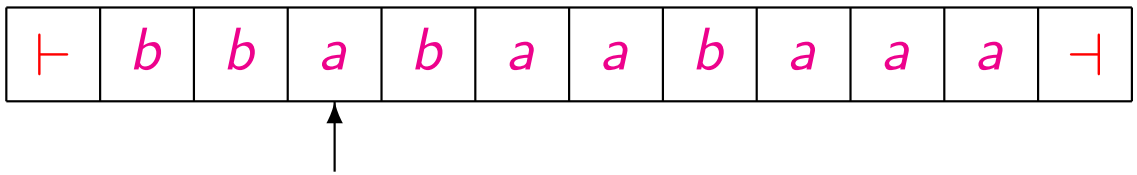
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

**finché** non trovi una $a$ muovi a destra

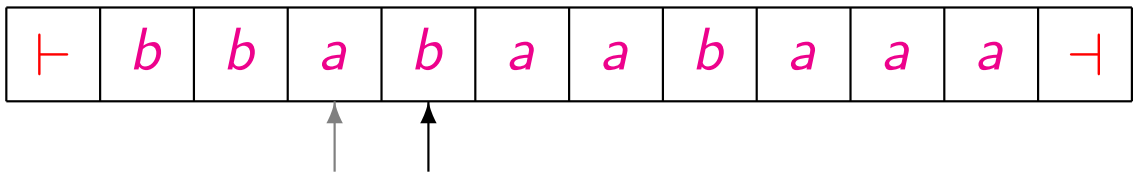
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

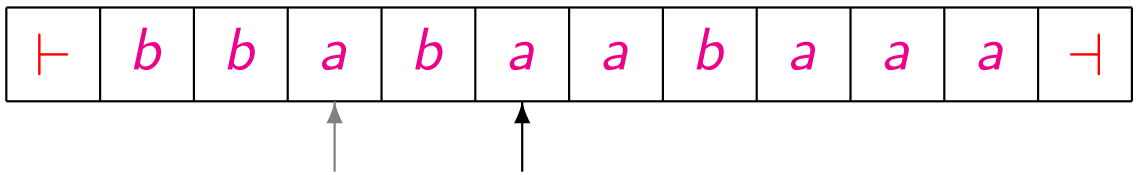



$n = 4$

**finché** non trovi una $a$ muovi a destra

muovi $n$ celle a destra

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

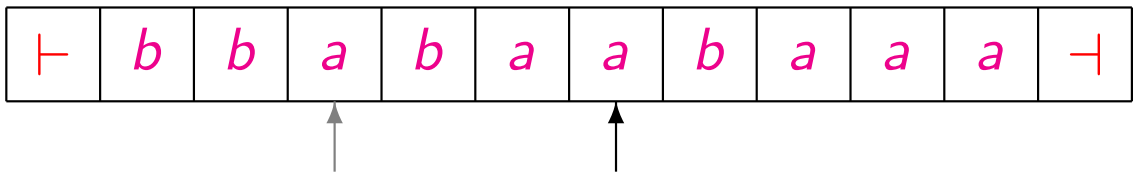



$n = 4$

**finché** non trovi una $a$ muovi a destra

muovi $n$ celle a destra

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

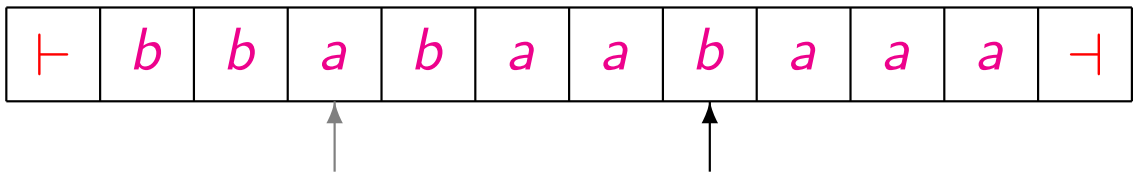



$n = 4$

**finché** non trovi una $a$ muovi a destra

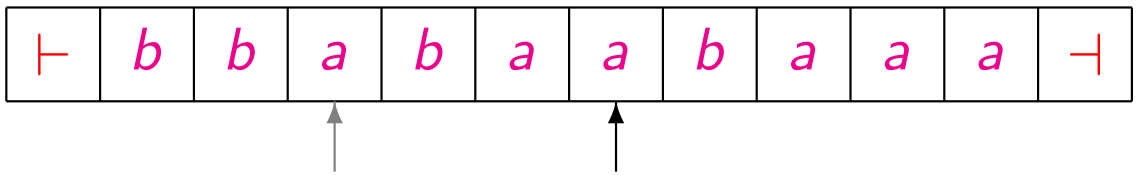
muovi $n$ celle a destra

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



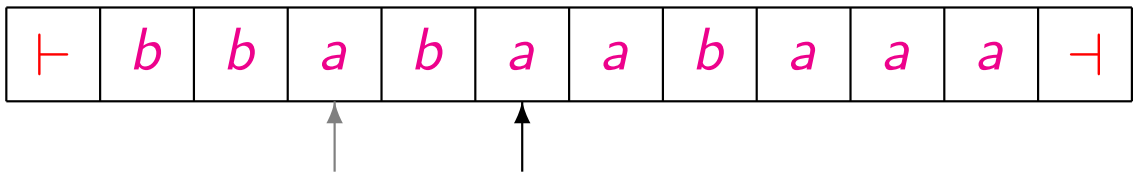

**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra
se c'è una $a$ **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



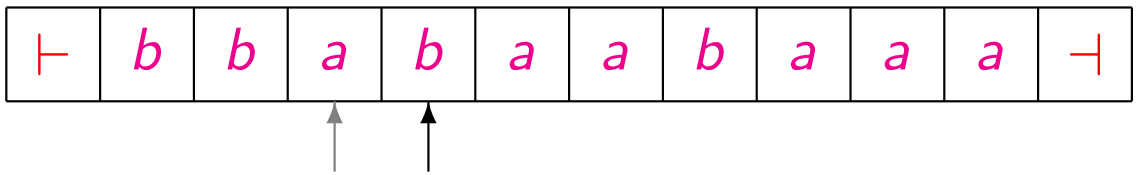

**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra
**se** c'è una $a$ **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra
**se** c'è una $a$ **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

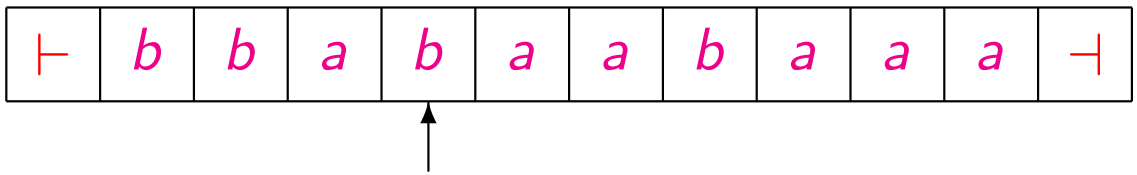
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

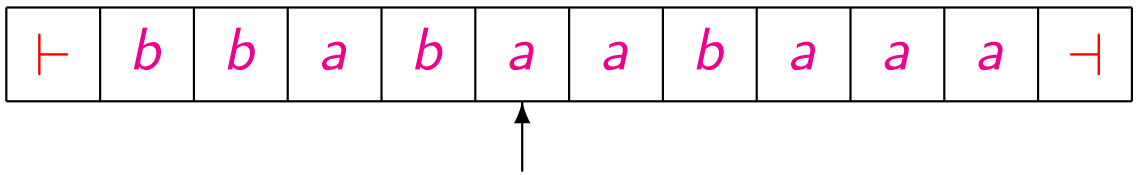
**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra
**se** c'è una $a$ **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




**finché** non trovi una *a* muovi a destra
muovi *n* celle a destra
**se** c'è una *a* **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

**finché** non trovi una $a$ muovi a destra

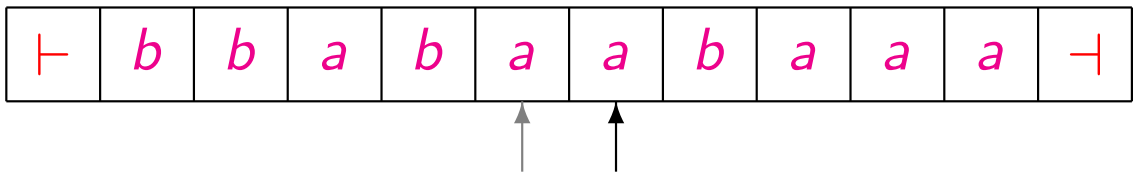
muovi $n$ celle a destra

**se** c'è una $a$ **allora accetta**

**altrimenti** muovi $n-1$ celle a sinistra

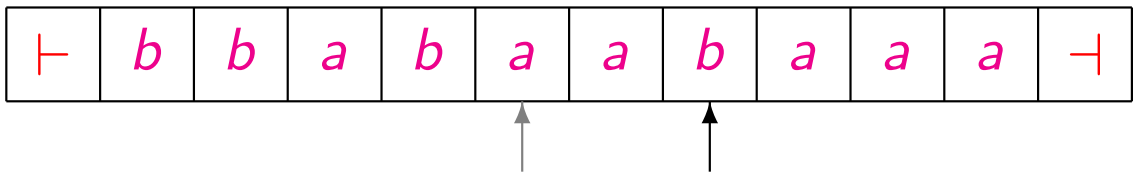
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



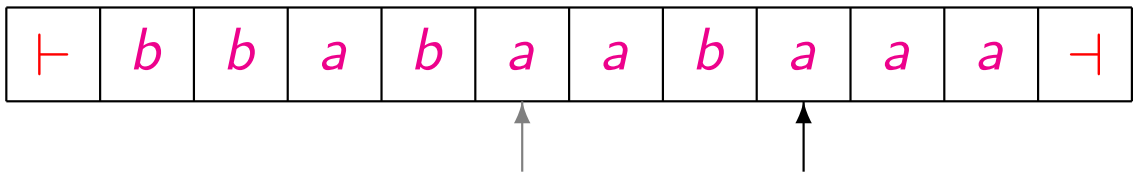

**finché** non trovi una *a* muovi a destra
muovi *n* celle a destra
**se** c'è una *a* **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$



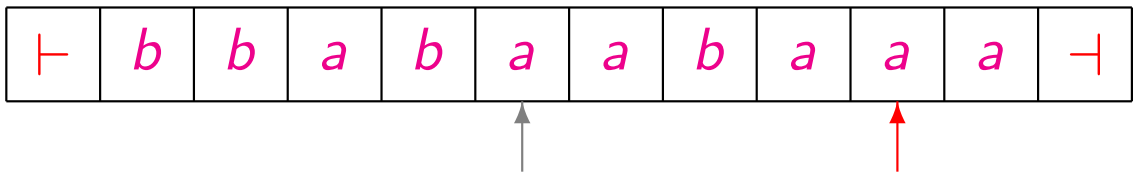

**finché** non trovi una *a* muovi a destra
muovi *n* celle a destra
**se** c'è una *a* **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




**finché** non trovi una *a* muovi a destra
muovi *n* celle a destra
**se** c'è una *a* **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

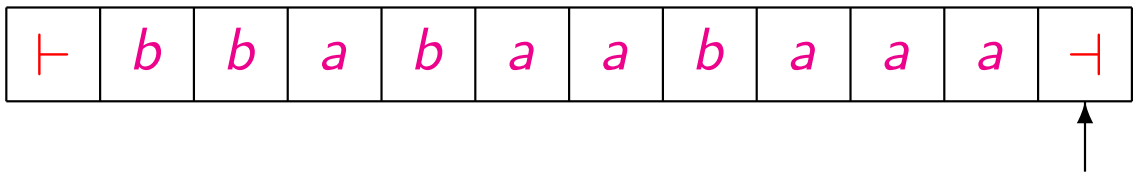
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




$n = 4$

**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra
**se** c'è una $a$ **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$




**finché** non trovi una *a* muovi a destra
muovi *n* celle a destra
**se** c'è una *a* **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo
*Eccezione:* **se** raggiungi ⊣ **allora rifiuta**

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

| $\vdash$ | $b$ | $b$ | $a$ | $b$ | $a$ | $a$ | $b$ | $a$ | $a$ | $a$ | $\dashv$ |
|---|---|---|---|---|---|---|---|---|---|---|---|

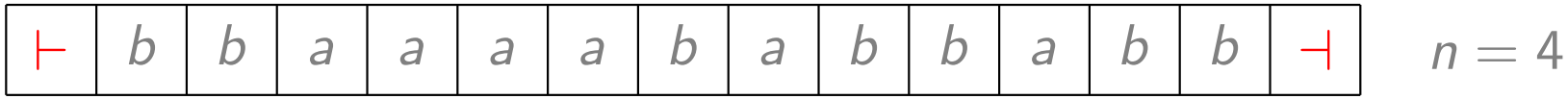
$n = 4$

**finché** non trovi una $a$ muovi a destra
muovi $n$ celle a destra
**se** c'è una $a$ **allora accetta**
**altrimenti** muovi $n-1$ celle a sinistra
ricomincia dal primo passo
*Eccezione:* **se** raggiungi $\dashv$ **allora rifiuta**

2DFA: $2n+\ldots$ stati

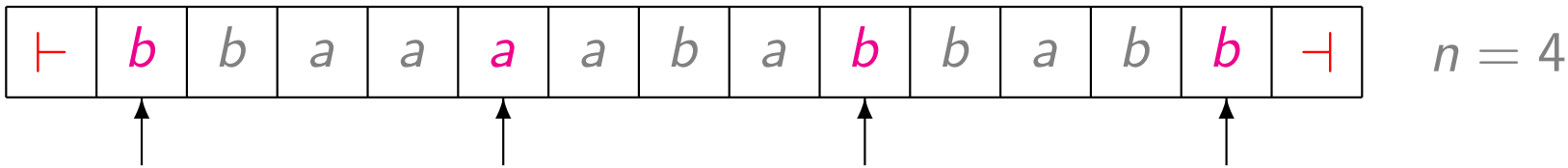
# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

## Un algoritmo differente



# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

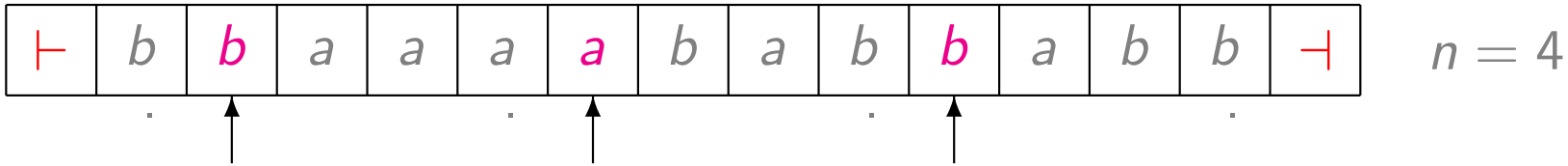
## Un algoritmo differente




Controlla tutte le posizioni $k$ con $k \equiv 1 \pmod{n}$

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

## Un algoritmo differente

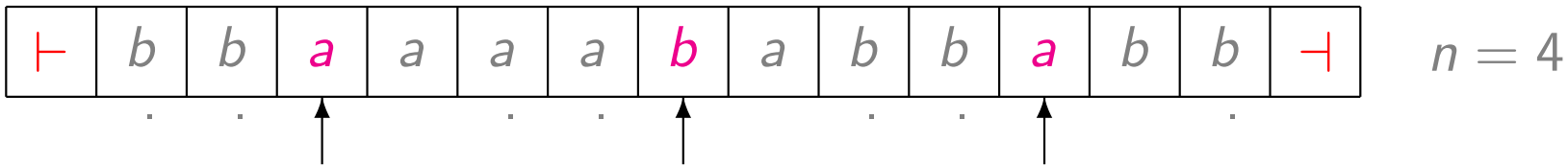



Controlla tutte le posizioni $k$ con $k \equiv 1 \pmod{n}$

Controlla tutte le posizioni $k$ con $k \equiv 2 \pmod{n}$

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

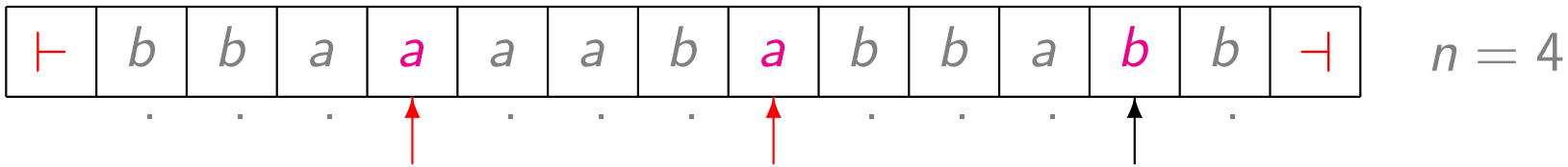
## Un algoritmo differente




Controlla tutte le posizioni $k$ con $k \equiv 1 \pmod{n}$
Controlla tutte le posizioni $k$ con $k \equiv 2 \pmod{n}$
...

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

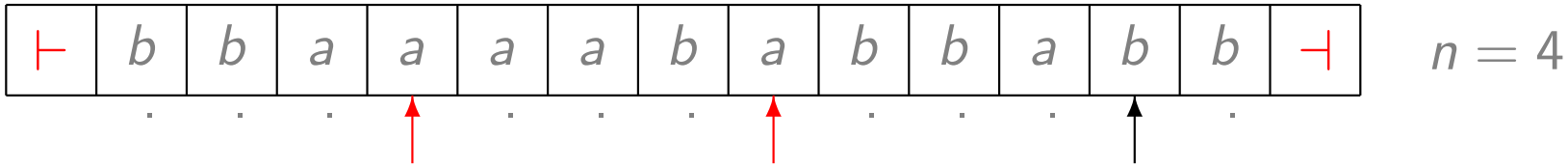
## Un algoritmo differente




Controlla tutte le posizioni $k$ con $k \equiv 1 \pmod{n}$
Controlla tutte le posizioni $k$ con $k \equiv 2 \pmod{n}$
...
Controlla tutte le posizioni $k$ con $k \equiv n \pmod{n}$

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

## Un algoritmo differente

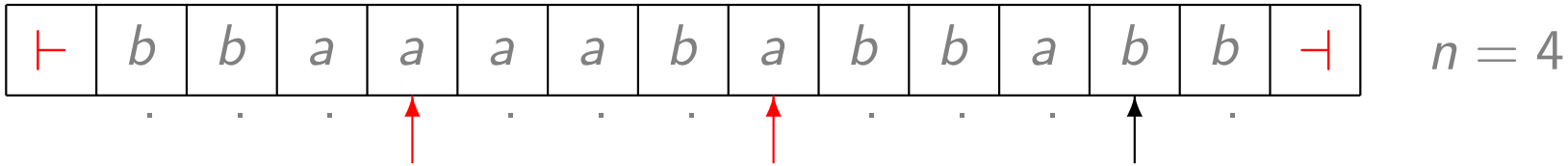



Controlla tutte le posizioni $k$ con $k \equiv 1 \pmod{n}$
Controlla tutte le posizioni $k$ con $k \equiv 2 \pmod{n}$
...
Controlla tutte le posizioni $k$ con $k \equiv n \pmod{n}$

Questa strategia può essere implementata usando $O(n)$ stati!

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

## Un algoritmo differente




Controlla tutte le posizioni $k$ con $k \equiv 1 \pmod{n}$
Controlla tutte le posizioni $k$ con $k \equiv 2 \pmod{n}$
...
Controlla tutte le posizioni $k$ con $k \equiv n \pmod{n}$

Questa strategia può essere implementata usando $O(n)$ stati!

Automi *sweeping*:

- Transizioni deterministiche
- Inversioni *solo alle estremità* dell'input

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Dunque:

- $L_n$ è accettato da
  - 1NFA
  - 2DFA
  - automa sweeping

  con $O(n)$ stati
- Ogni 1DFA richiede un numero di stati esponenziale

*Anche in questo esempio,*
si può rimuovere il nondeterminismo utilizzando le inversioni
ottenendo un numero di stati lineare

**Problema**
È sempre possibile *sostituire nondeterminismo con inversioni*
senza aumentare di molto gli stati dell'automa?

# Esempio: $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Dunque:

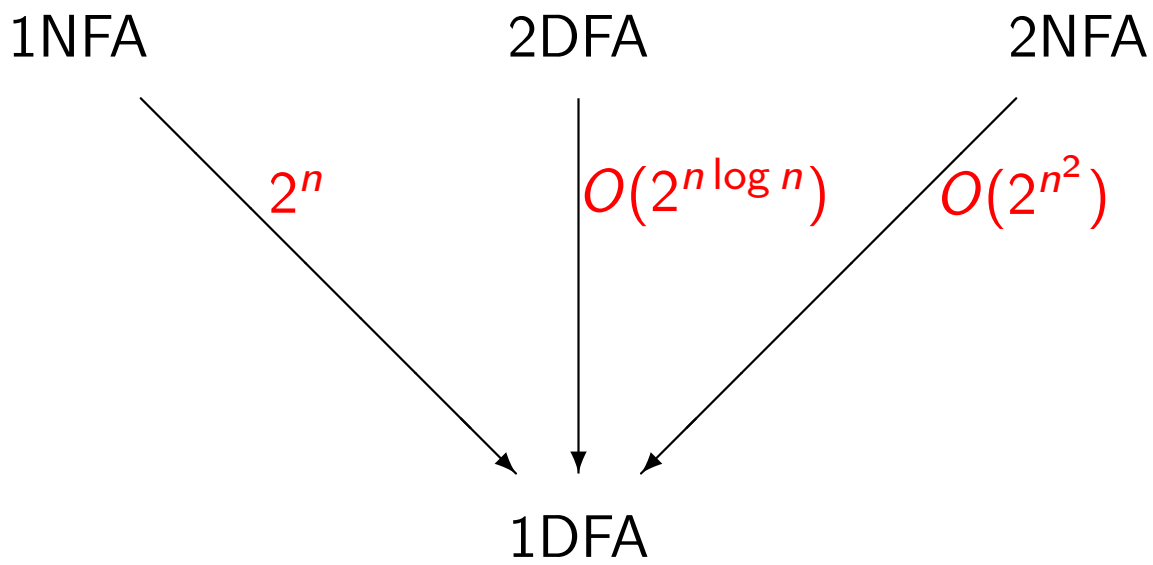
- $L_n$ è accettato da
  - 1NFA
  - 2DFA
  - automa sweeping

  con $O(n)$ stati
- Ogni 1DFA richiede un numero di stati esponenziale

*Anche in questo esempio,*
si può rimuovere il nondeterminismo utilizzando le inversioni ottenendo un numero di stati lineare

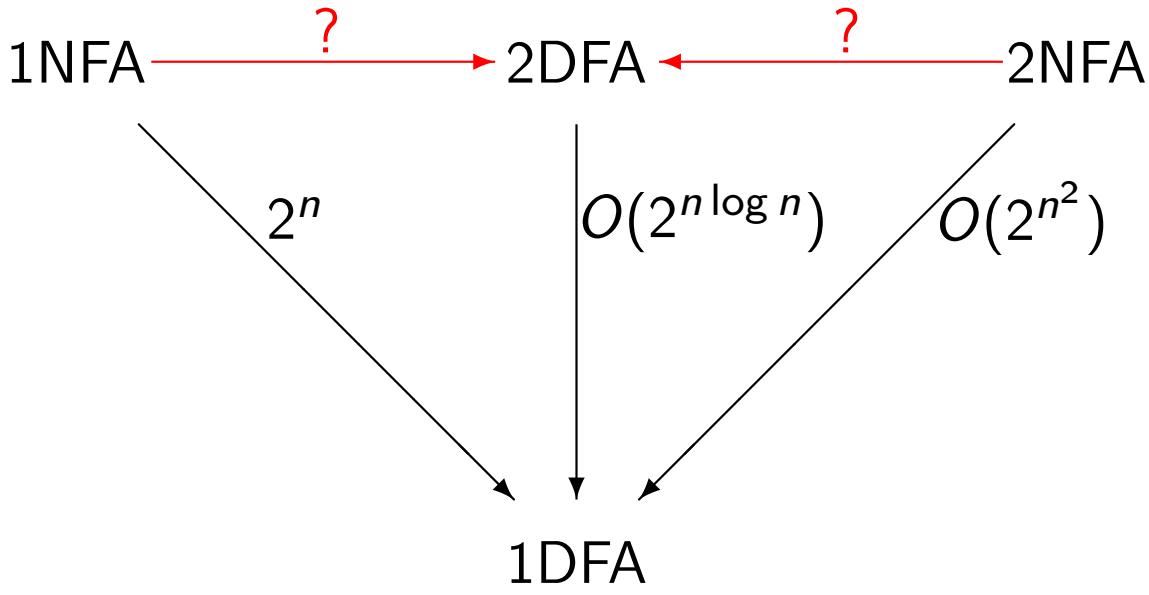
**Problema**
È sempre possibile *sostituire nondeterminismo con inversioni* senza aumentare di molto gli stati dell'automa?

# Costi delle simulazioni ottimali tra automi




[Rabin&Scott '59, Shepardson '59, Meyer&Fischer '71, . . . ]

# Costi delle simulazioni ottimali tra automi

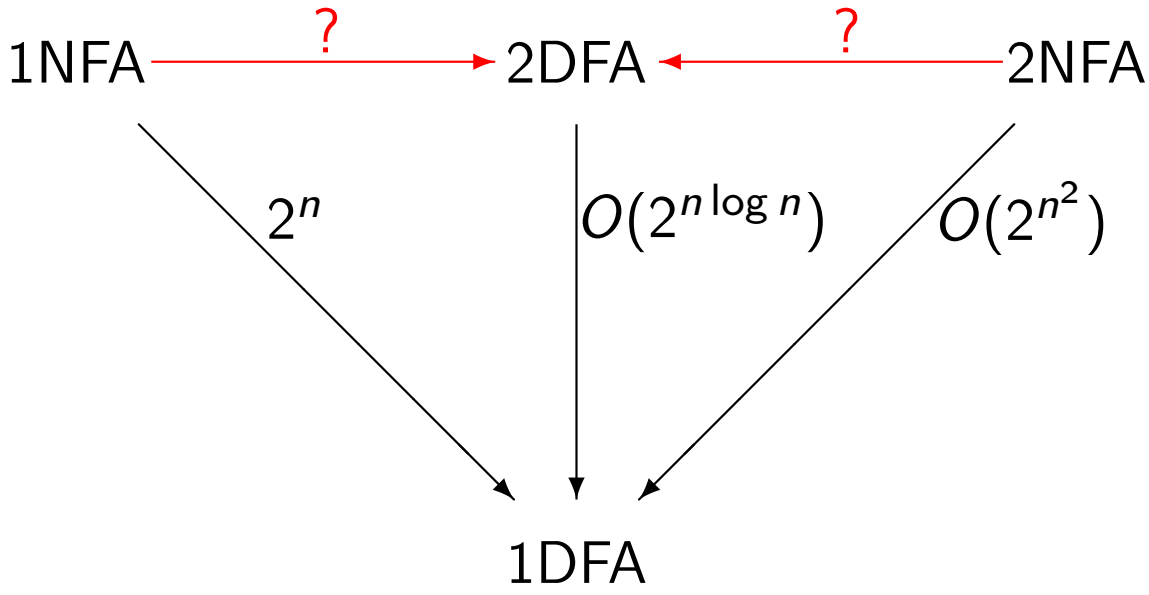



[Rabin&Scott '59, Shepardson '59, Meyer&Fischer '71, . . . ]

## Questione

*"Quanto" la possibilità di muovere la testina in entrambe le direzioni risulta utile nell'eliminazione del nondeterminismo?*

# Costi delle simulazioni ottimali tra automi

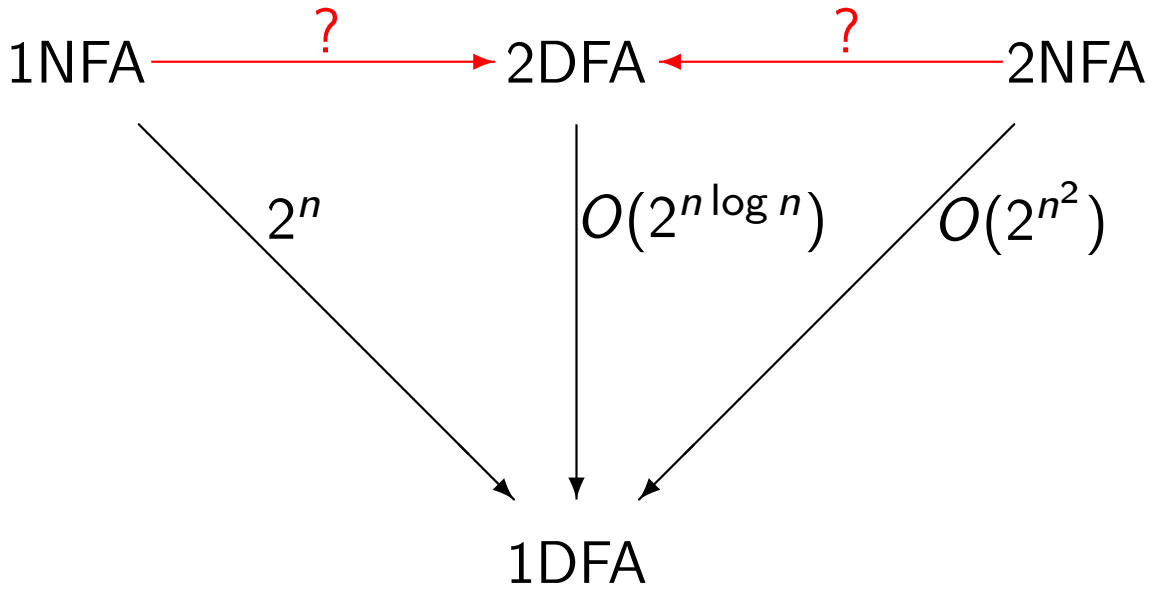



**Problema ([Sakoda&Sipser '78])**

*Esistono simulazioni polinomiali di*

- *1NFA mediante 2DFA*
- *2NFA mediante 2DFA ?*

# Costi delle simulazioni ottimali tra automi




## Problema ([Sakoda&Sipser '78])

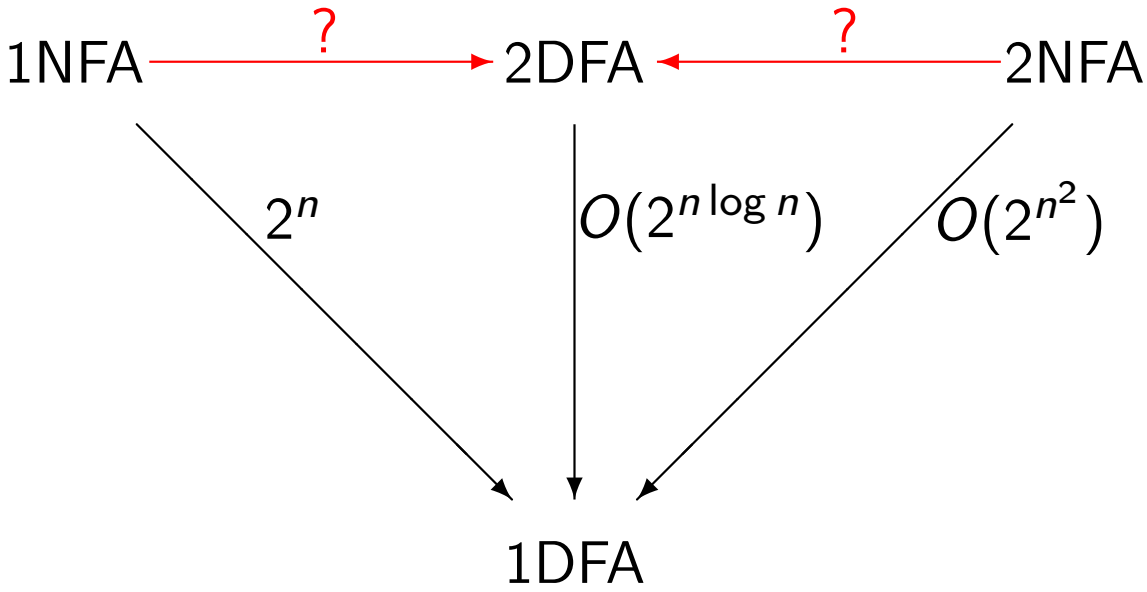
*Esistono simulazioni polinomiali di*

- *1NFA mediante 2DFA*
- *2NFA mediante 2DFA ?*

## Congettura

*Queste simulazioni non sono polinomiali*

# Costi delle simulazioni ottimali tra automi




- Limiti superiori esponenziali
  dalle simulazioni di 1NFA e 2NFA mediante 1DFA
- Limiti inferiori polinomiali
  $\Omega(n^2)$ per la simulazione di 1NFA mediante 2DFA

[Chrobak '86]

# Il problema di Sakoda e Sipser

- Molto difficile nel caso generale
- Risultati poco incoraggianti:

  Limiti inferiori e superiori molto distanti
  (Polinomiali vs esponenziali)

- Dunque:

  Studio di versioni particolari del problema!

# NFA vs 2DFA: versioni particolari

(i) Restrizioni sulle macchine simulanti (2DFA)

- automi *sweeping* [Sipser '80]
- automi *oblivious* [Hromkovič&Schnitger '03]
- automi con *poche inversioni* [Kapoutsis '11]

(ii) Restrizioni sui linguaggi

- *caso unario* [Geffert Mereghetti&P '03]

(iii) Restrizioni sulle macchine simulate (2NFA)

- automi *outer nondeterministic* [Guillon Geffert&P '12]

# NFA vs 2DFA: versioni particolari

(i) Restrizioni sulle macchine simulanti (2DFA)

- automi *sweeping* [Sipser '80]
- automi *oblivious* [Hromkovič&Schnitger '03]
- automi con *poche inversioni* [Kapoutsis '11]

(ii) Restrizioni sui linguaggi

- *caso unario* [Geffert Mereghetti&P '03]

(iii) Restrizioni sulle macchine simulate (2NFA)

- automi *outer nondeterministic* [Guillon Geffert&P '12]

# NFA vs 2DFA: versioni particolari

(i) Restrizioni sulle macchine simulanti (2DFA)

- automi *sweeping* [Sipser '80]
- automi *oblivious* [Hromkovič&Schnitger '03]
- automi con *poche inversioni* [Kapoutsis '11]

(ii) Restrizioni sui linguaggi

- *caso unario* [Geffert Mereghetti&P '03]

(iii) Restrizioni sulle macchine simulate (2NFA)

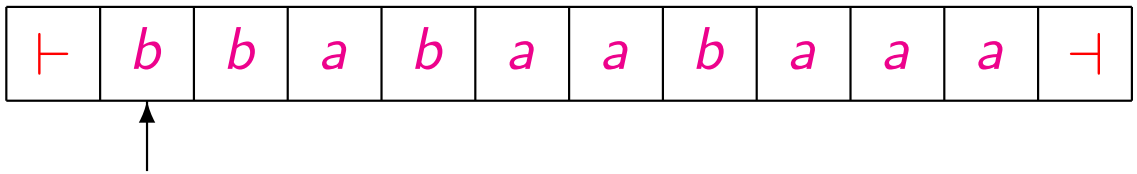
- automi *outer nondeterministic* [Guillon Geffert&P '12]

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

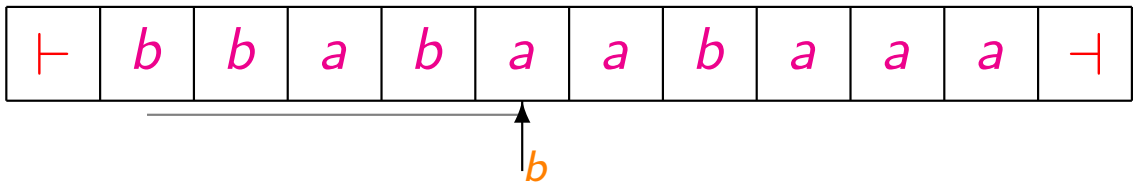
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

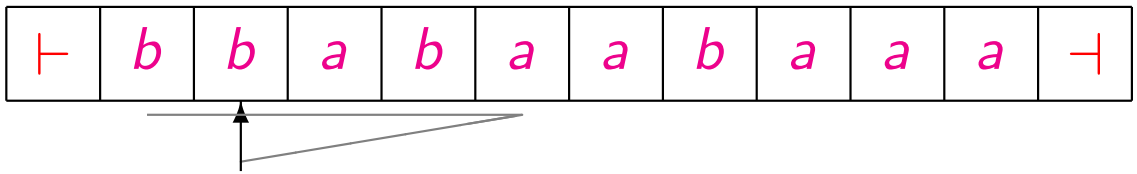
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

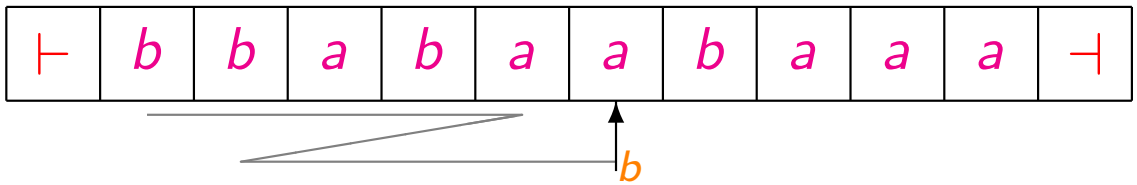
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

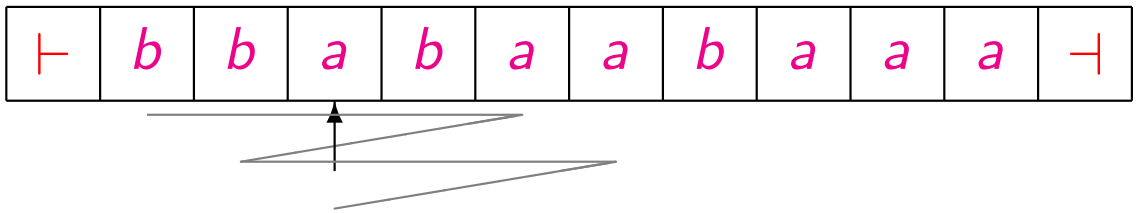
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

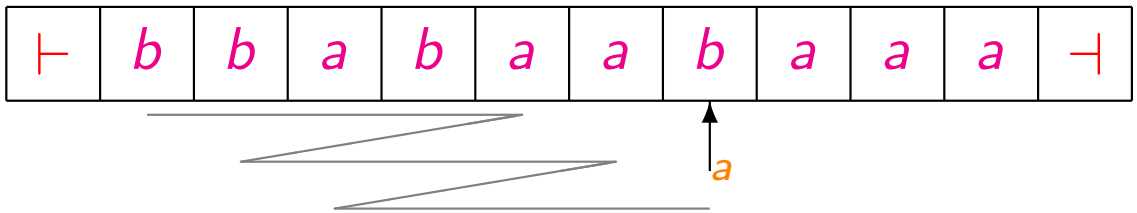
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

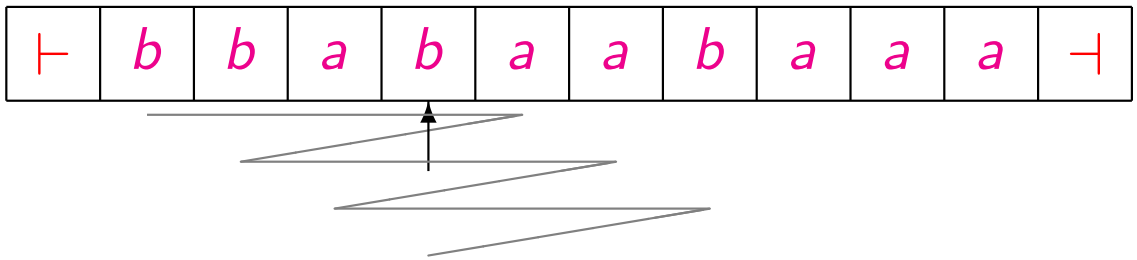
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

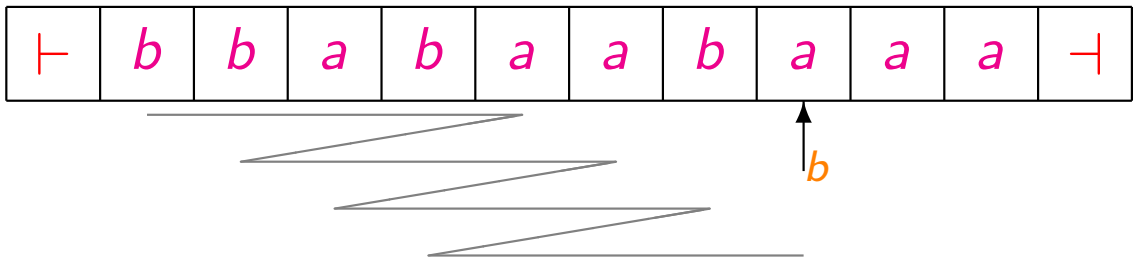
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

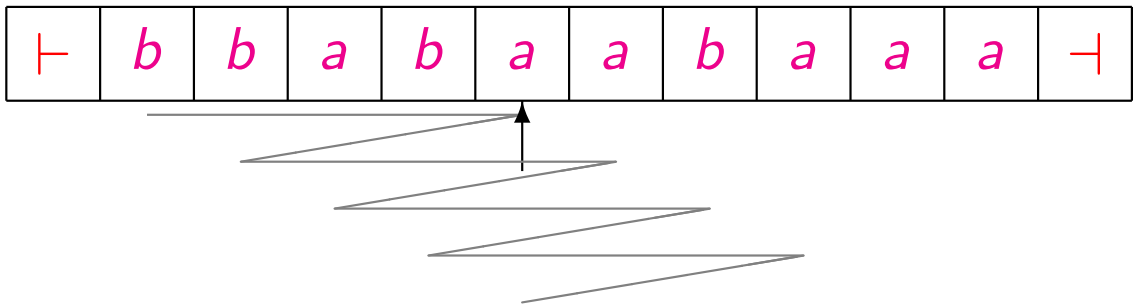
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




$n = 4$

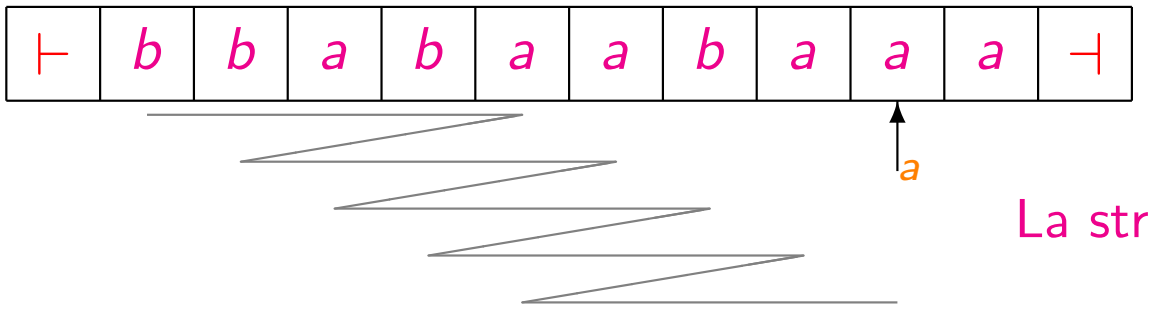
# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$



# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$

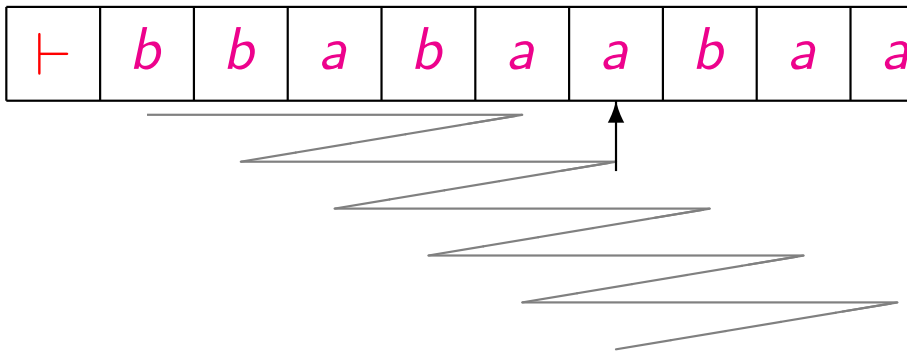



$n = 4$

La stringa può essere accettata!

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$

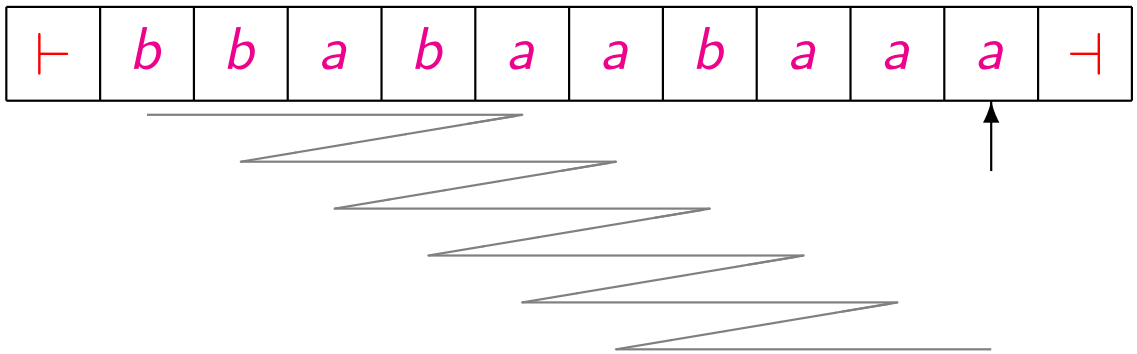



$n = 4$

La stringa può essere accettata!
...ma questo automa prosegue le scansioni

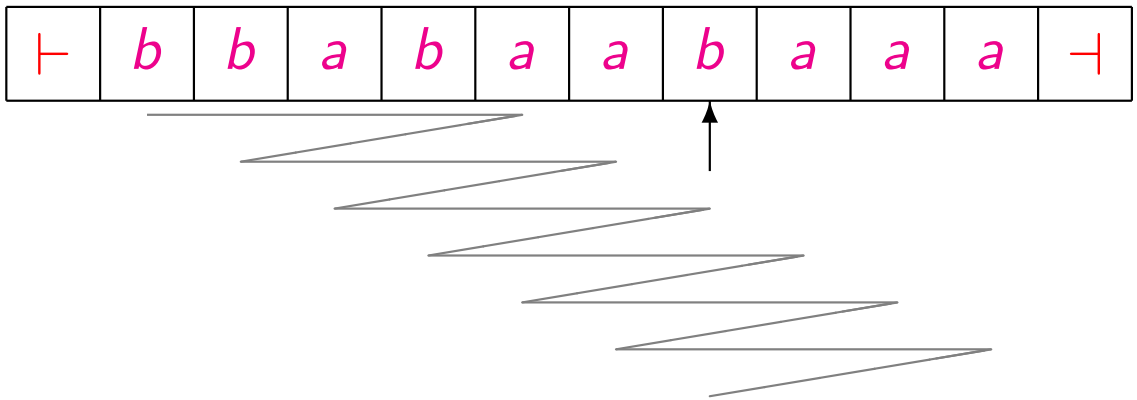
# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$



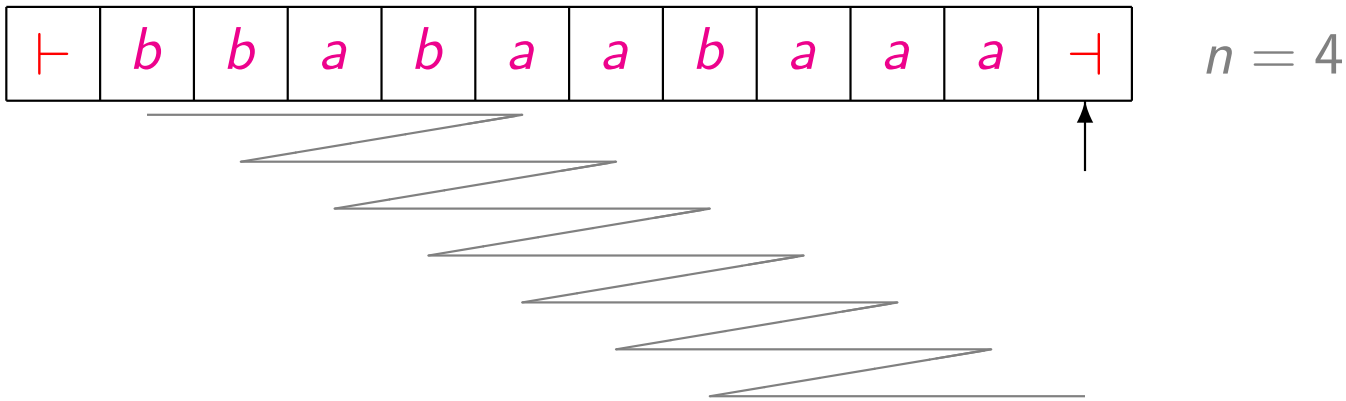
# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$



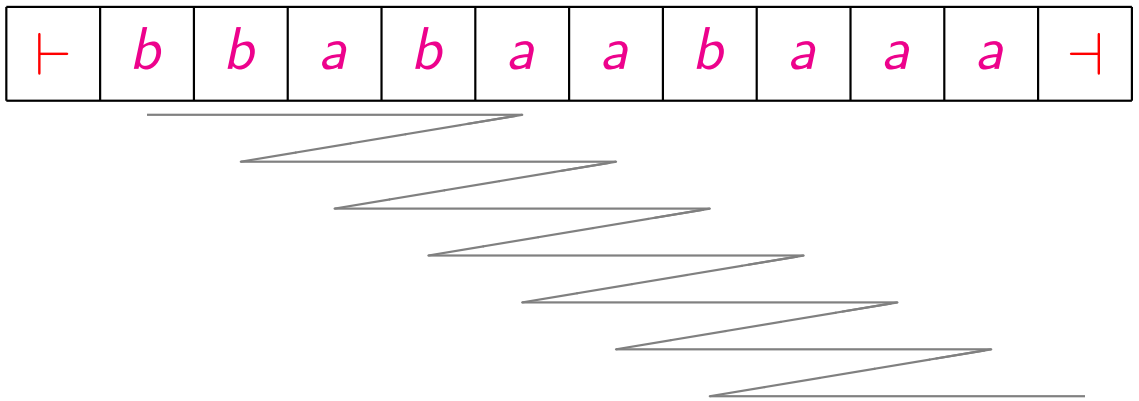
# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$



# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

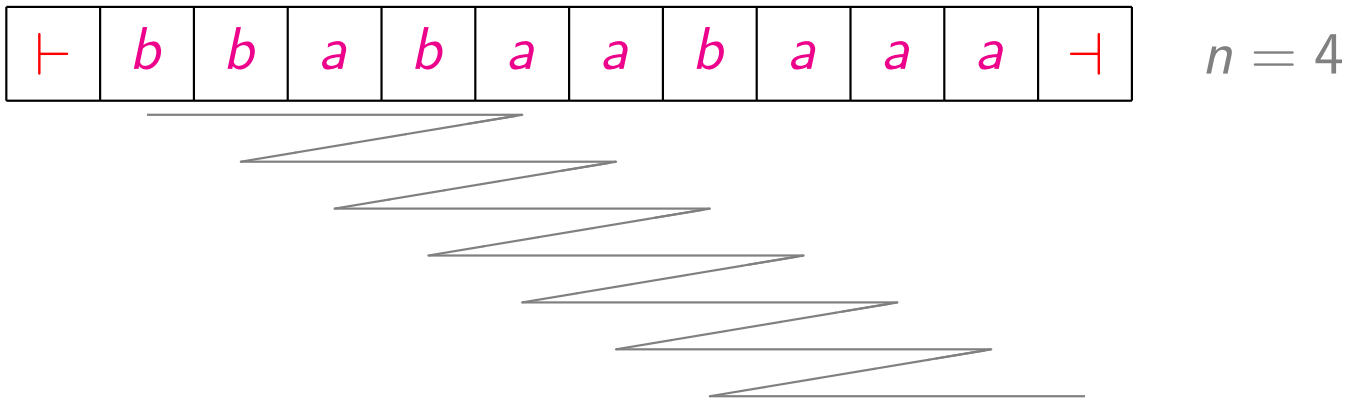
Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




Anche in questo caso $O(n)$ stati!

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$



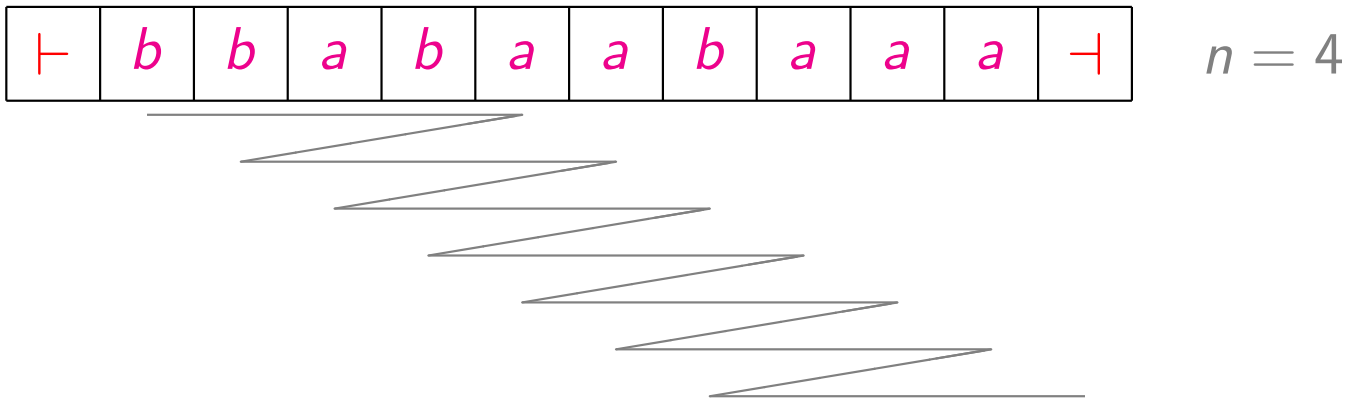

Anche in questo caso $O(n)$ stati!

*Automi oblivious:*

- Transizioni deterministiche
- Stessa "traiettoria" della testina per tutti gli input della stessa lunghezza

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$

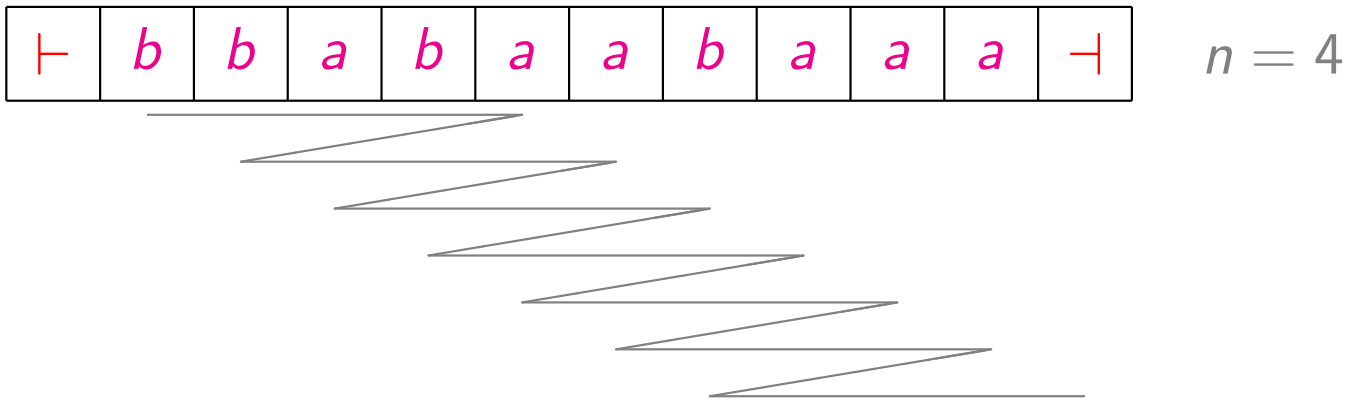



*Numero di inversioni della testina*:
Su input di lunghezza $m$:

- circa $2m$ inversioni,
  numero *lineare* rispetto alla lunghezza dell'input
- l'algoritmo "sweeping" ne utilizza circa $2n$,
  numero *costante* rispetto alla lunghezza dell'input

# Di nuovo $L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

Algoritmo "naif": confronta le posizioni $i$ e $i+n$, $i = 1, 2, \ldots$




*Numero di inversioni della testina*:
Su input di lunghezza $m$:

- circa $2m$ inversioni,
  numero *lineare* rispetto alla lunghezza dell'input
- l'algoritmo "sweeping" ne utilizza circa $2n$,
  numero *costante* rispetto alla lunghezza dell'input

# Un'altra possibile restrizione

*Automi con "poche" inversioni* [Kapoutsis '11]:

- Numero di inversioni sublineare rispetto alla lunghezza dell'input
- Modello *deterministico*

**Teorema ([Kapoutsis&P '12])**

*Se il numero di inversioni effettuate da un 2DFA è sublineare allora deve essere costante*

# Un'altra possibile restrizione

*Automi con "poche" inversioni* [Kapoutsis '11]:

- Numero di inversioni sublineare rispetto alla lunghezza dell'input
- Modello *deterministico*

**Teorema ([Kapoutsis&P '12])**

*Se il numero di inversioni effettuate da un 2DFA è sublineare allora deve essere costante*

# Separazioni

oblivious

sweeping

poche inversioni

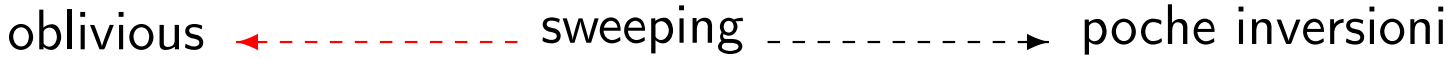
# Separazioni

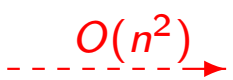
oblivious

sweeping $\dashrightarrow$ poche inversioni

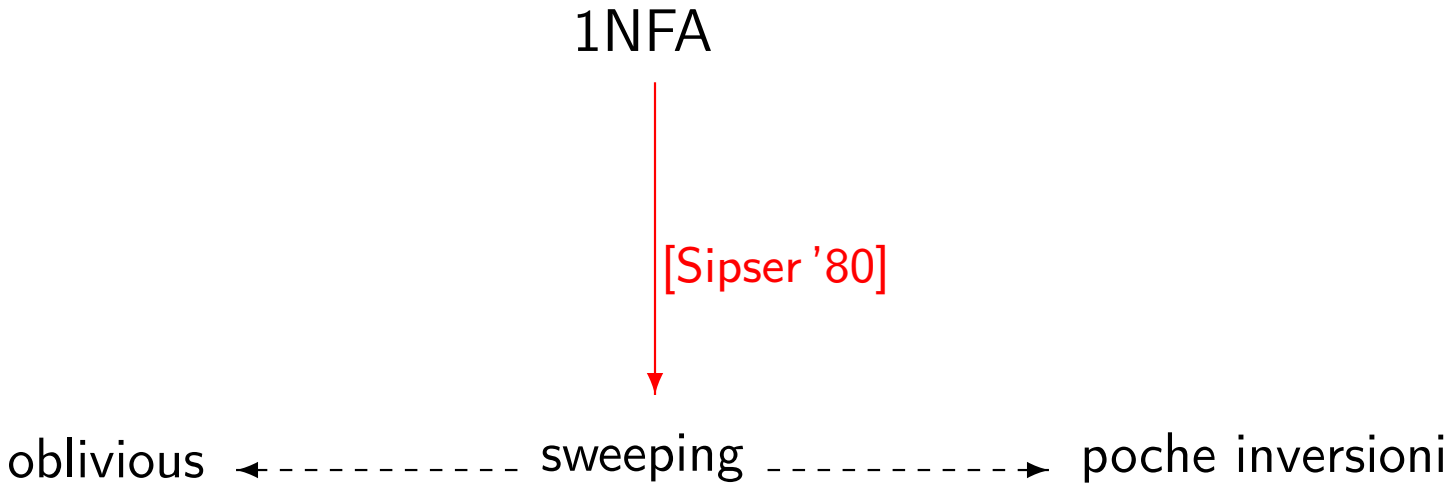
$\xrightarrow{O(n^2)}$

# Separazioni



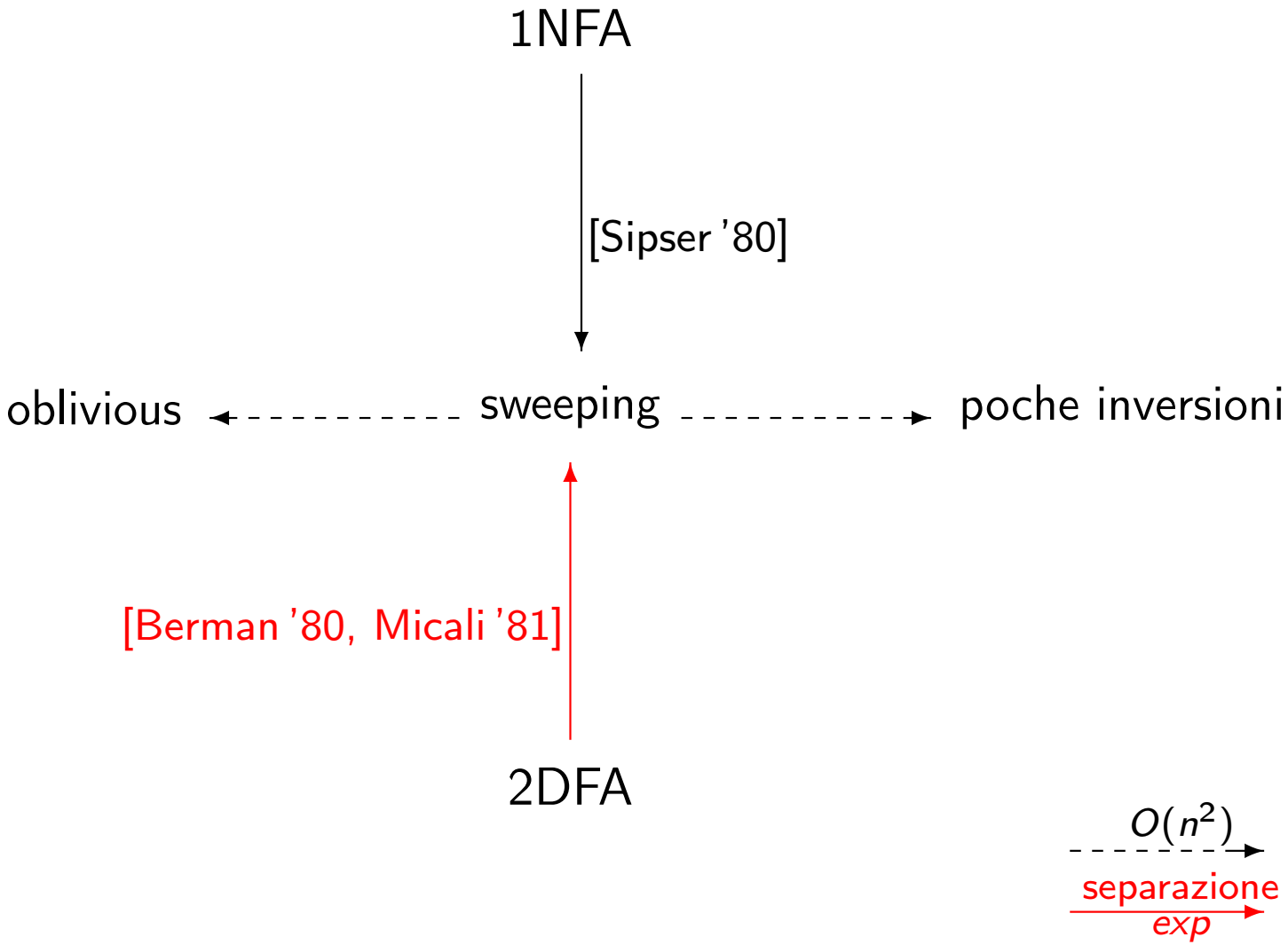
# Separazioni



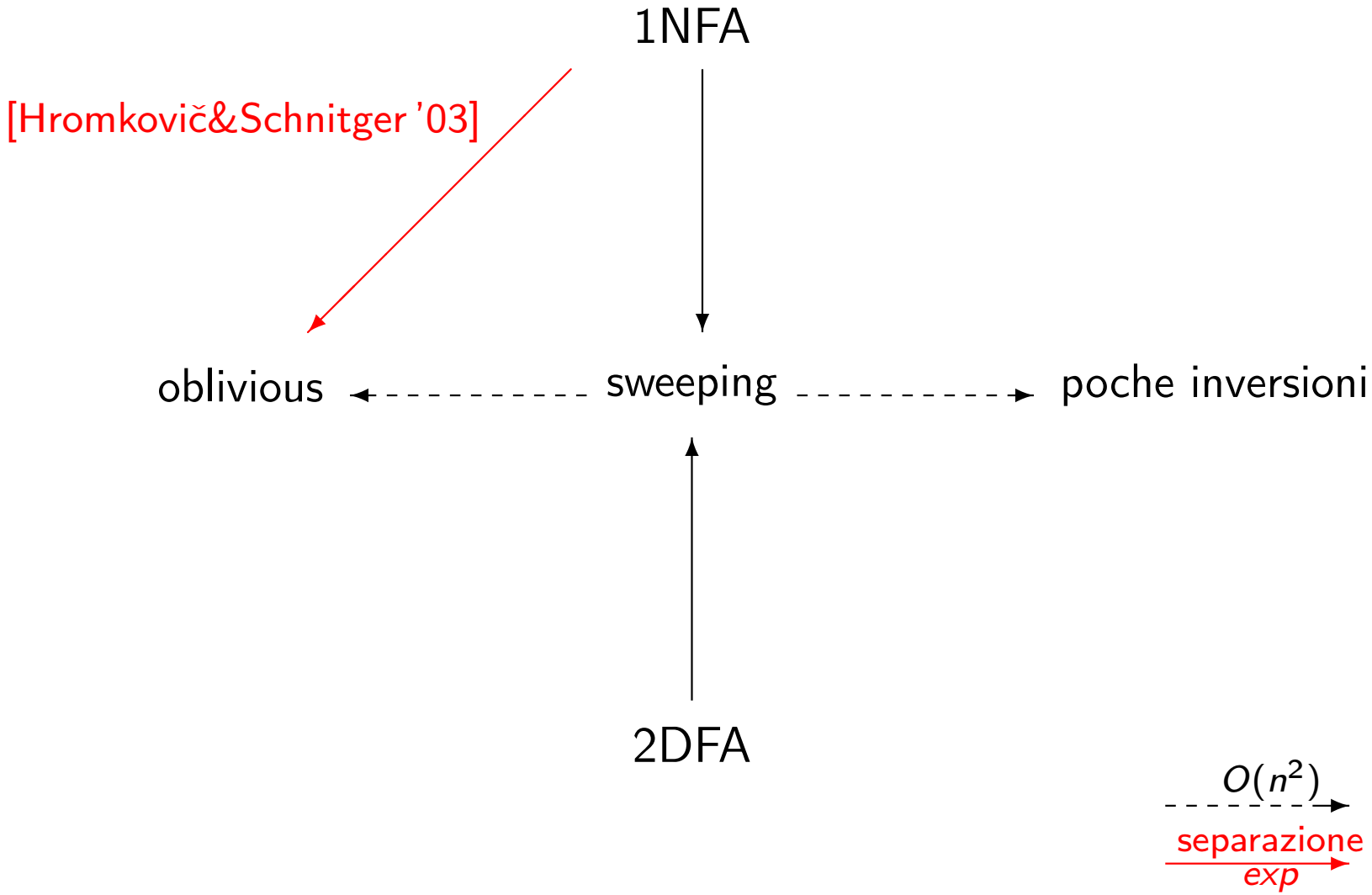

$O(n^2)$

separazione *exp*

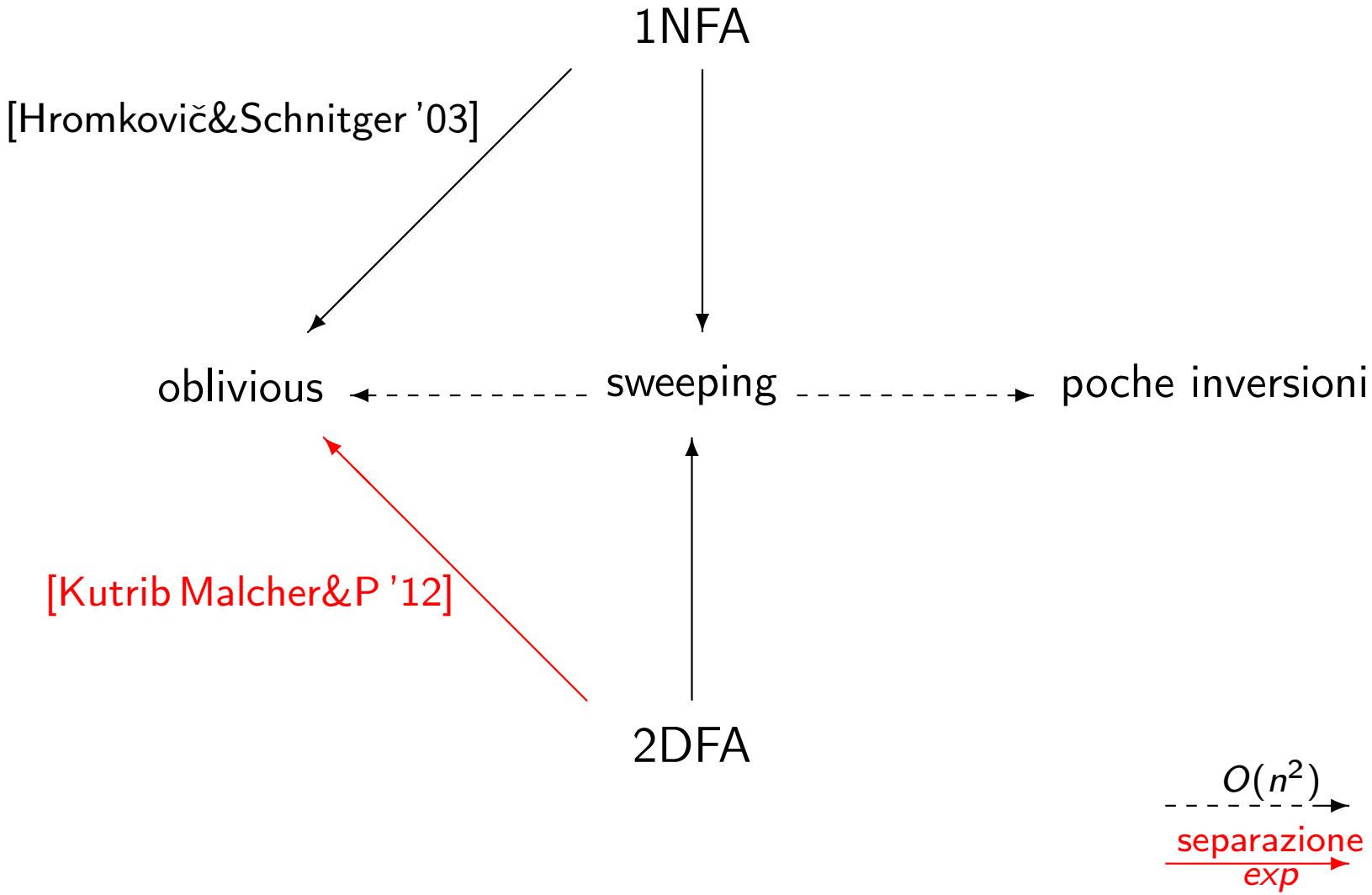
# Separazioni



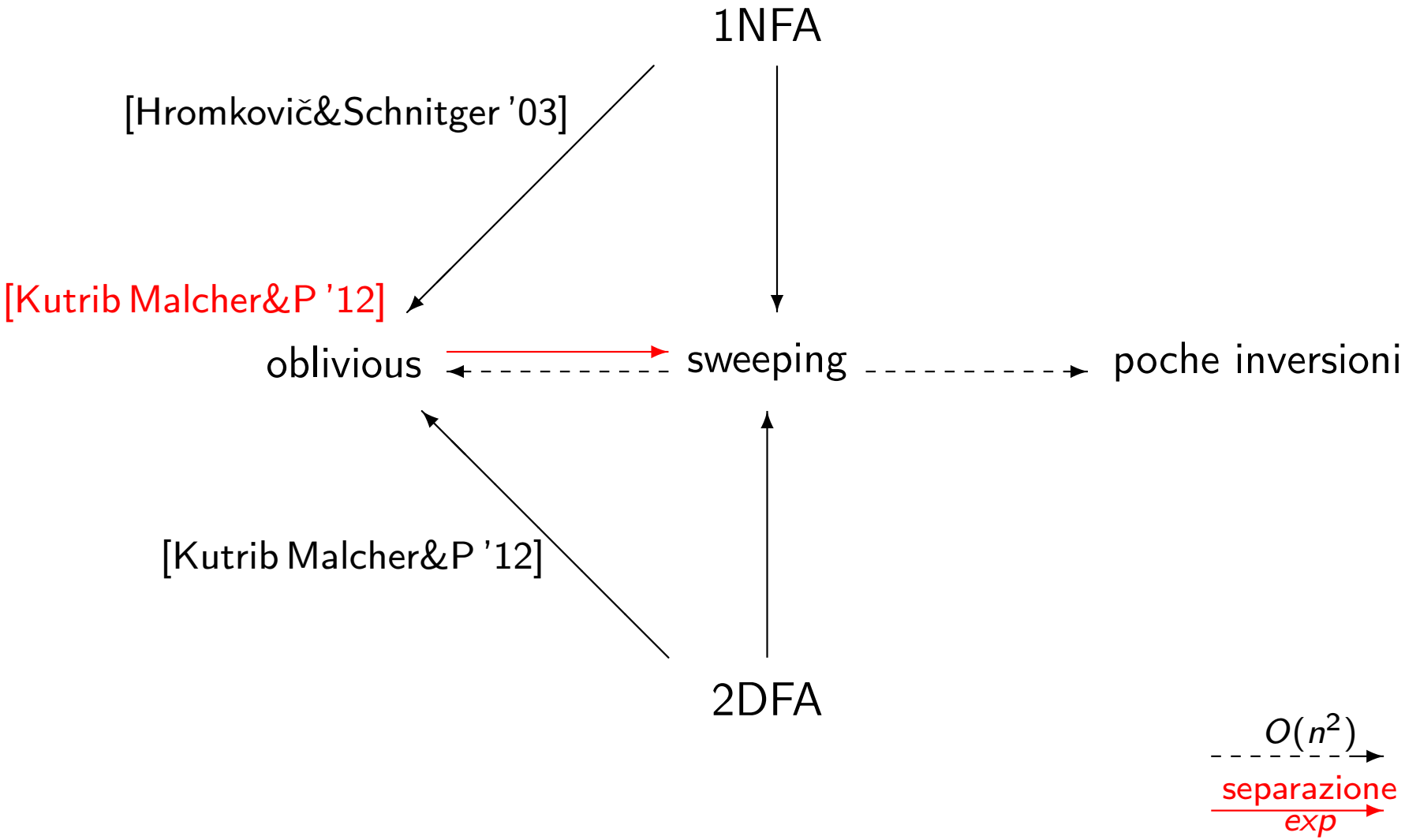
# Separazioni



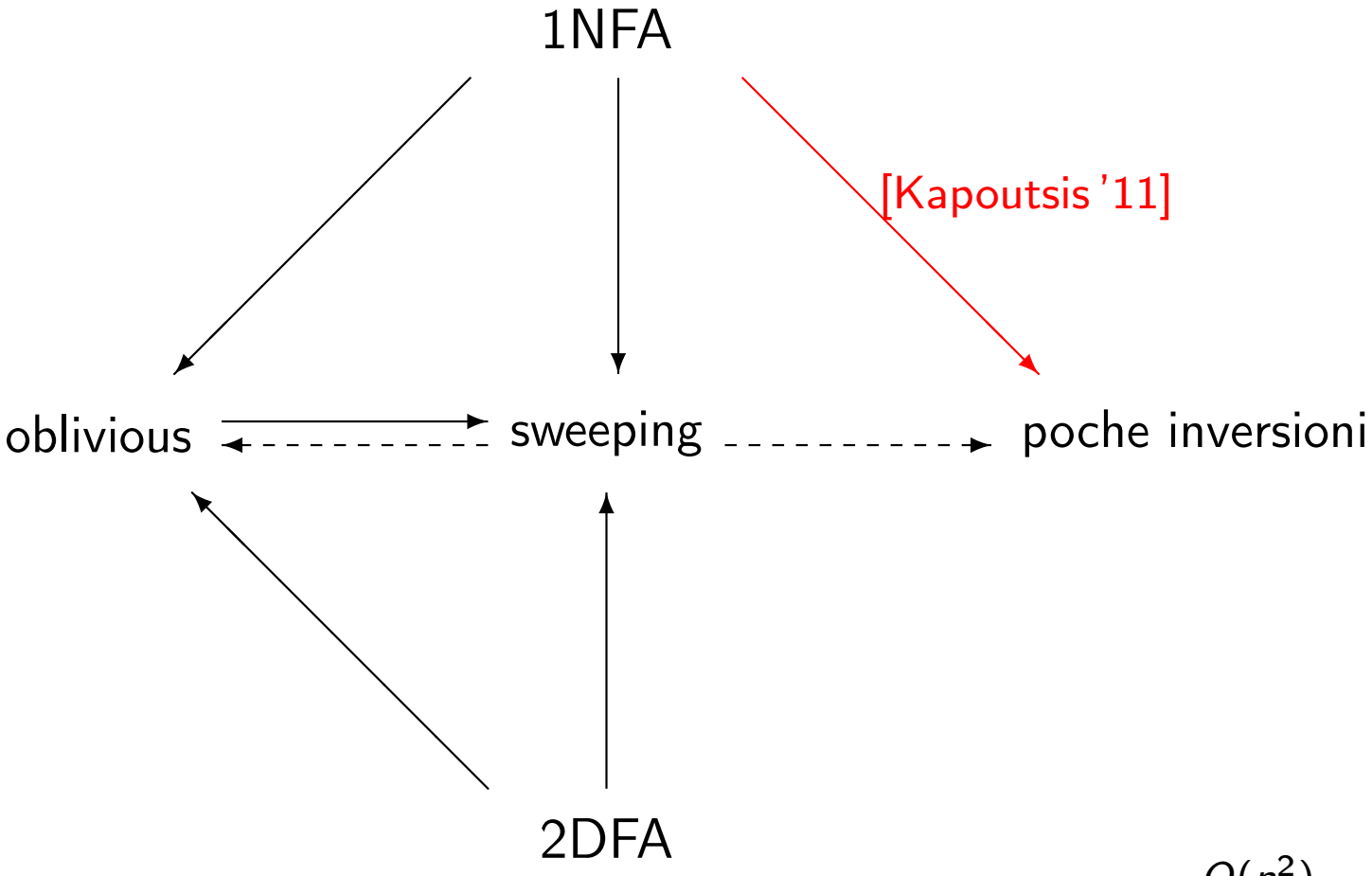
# Separazioni



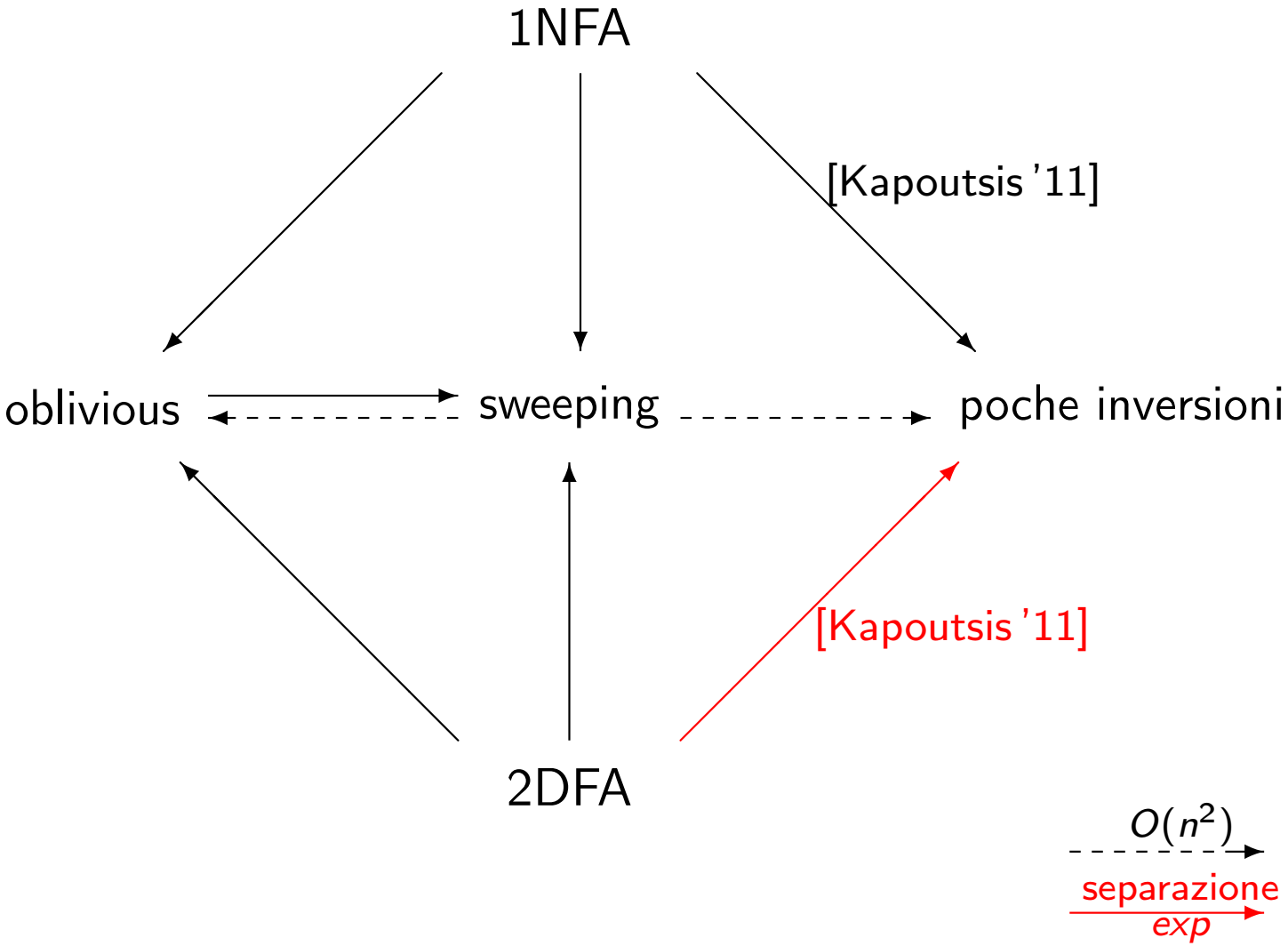
# Separazioni



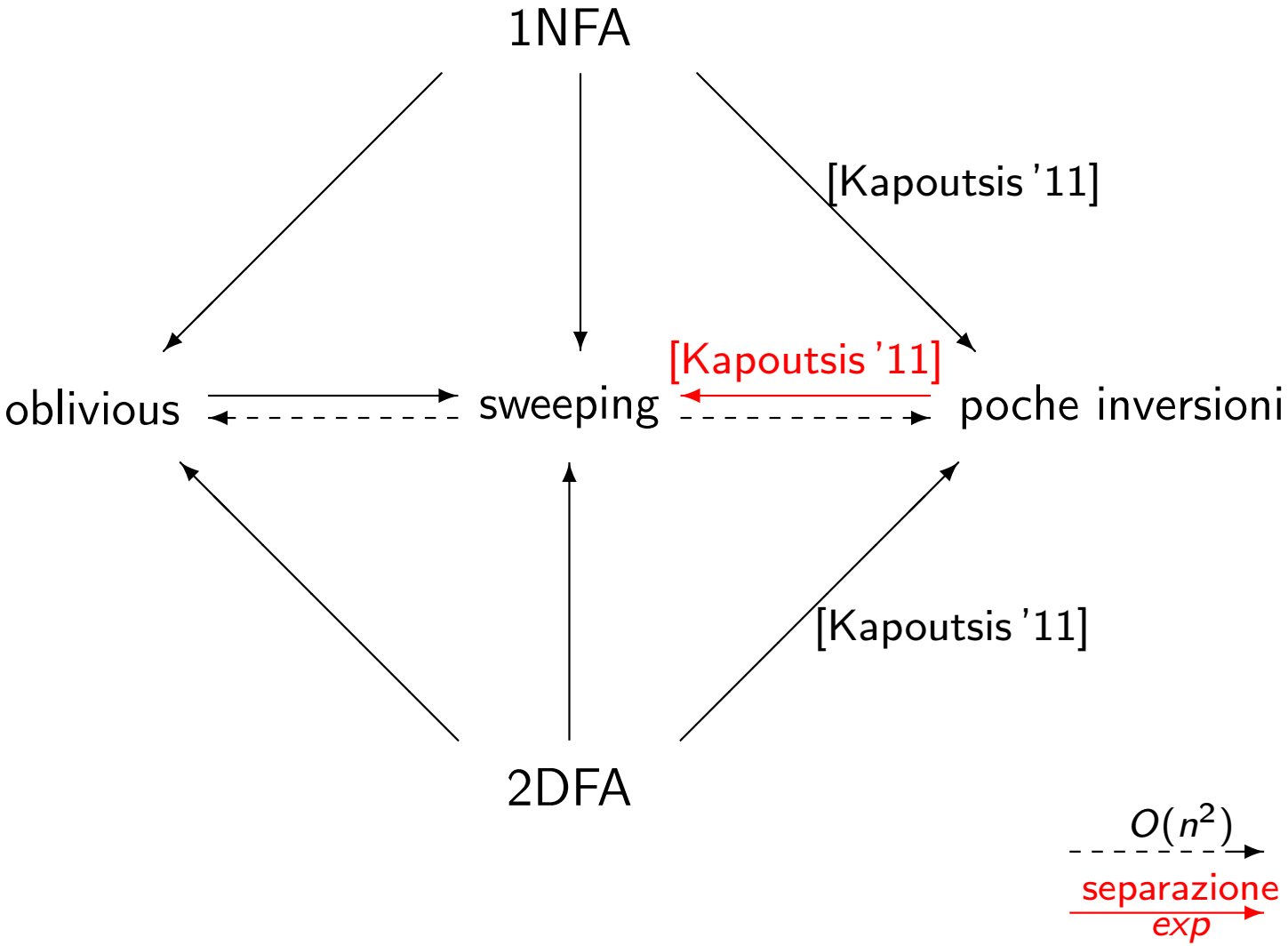
# Separazioni



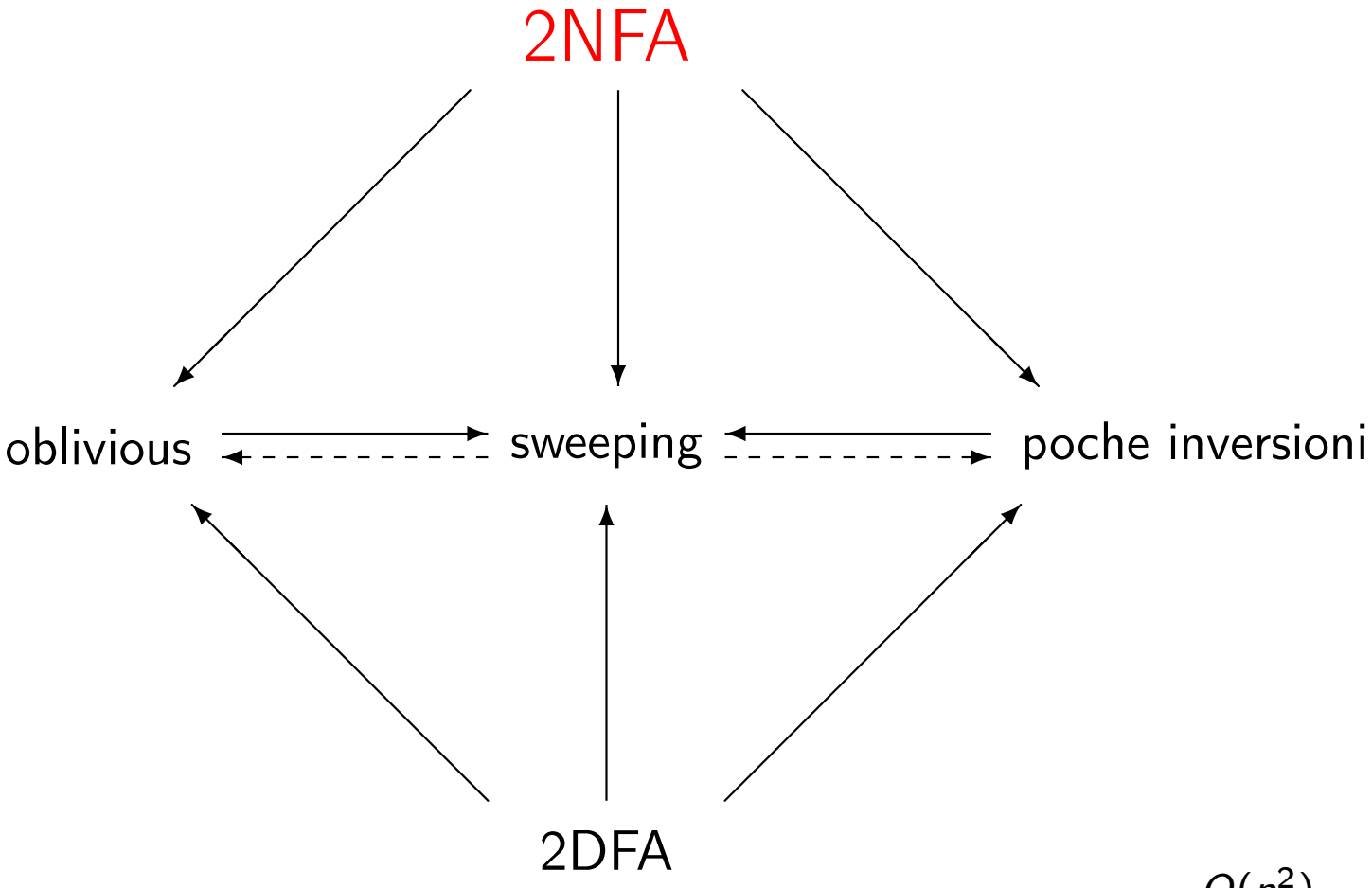
# Separazioni



# Separazioni



# Separazioni




$O(n^2)$

separazione
*exp*

# Il problema di Sakoda&Sipser

**Problema ([Sakoda&Sipser '78])**

*Esistono simulazioni polinomiali di*

- *1NFA mediante 2DFA*
- *2NFA mediante 2DFA ?*

Altra possibile restrizione:

Caso unario $\#\Sigma = 1$

# Simulazioni ottimali tra automi unari

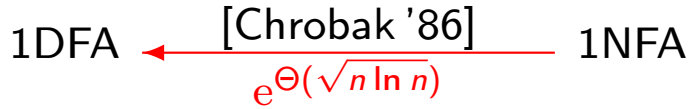
I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale

1DFA 1NFA

2DFA 2NFA

# Simulazioni ottimali tra automi unari

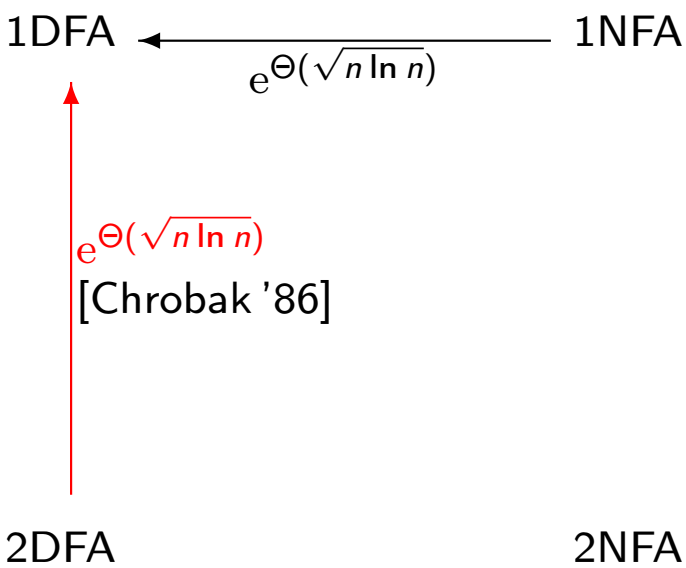
I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale




2DFA 2NFA

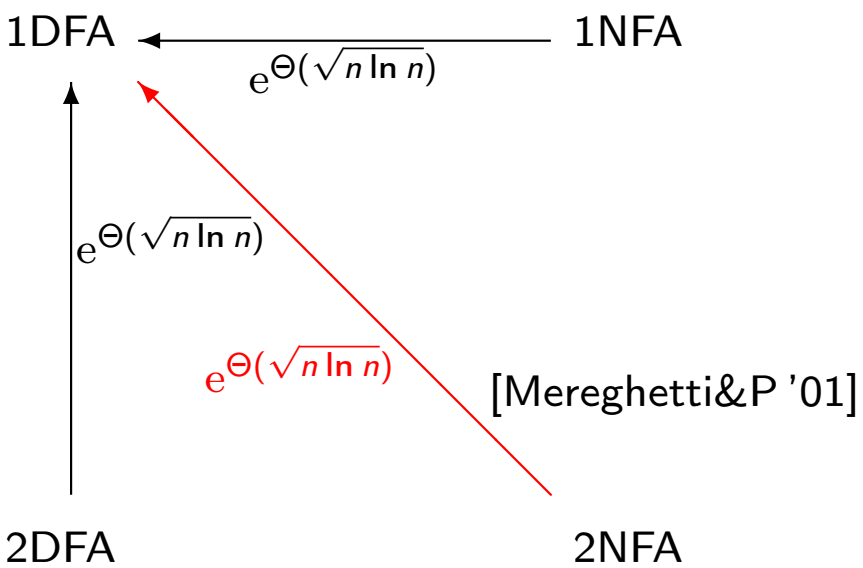
# Simulazioni ottimali tra automi unari

I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale



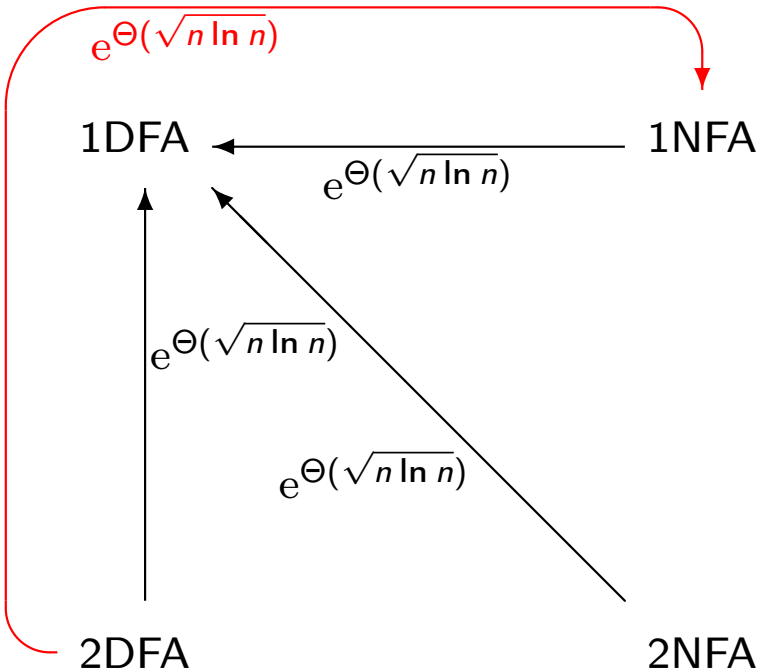
# Simulazioni ottimali tra automi unari

I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale



# Simulazioni ottimali tra automi unari

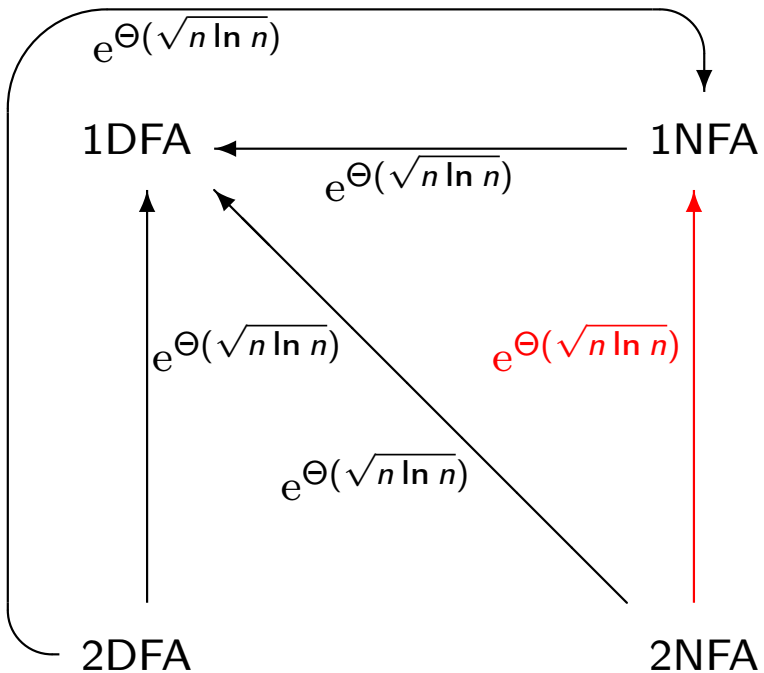
I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale




conseguenza di 2DFA $\rightarrow$ 1DFA

# Simulazioni ottimali tra automi unari

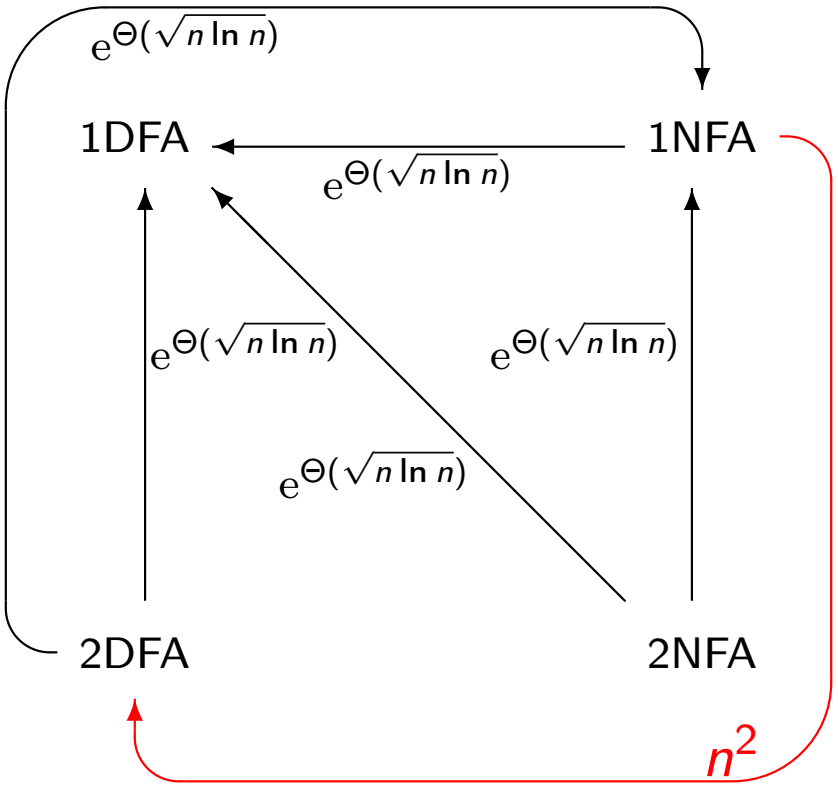
I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale




conseguenza di 2NFA $\rightarrow$ 1DFA

# Simulazioni ottimali tra automi unari

I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale

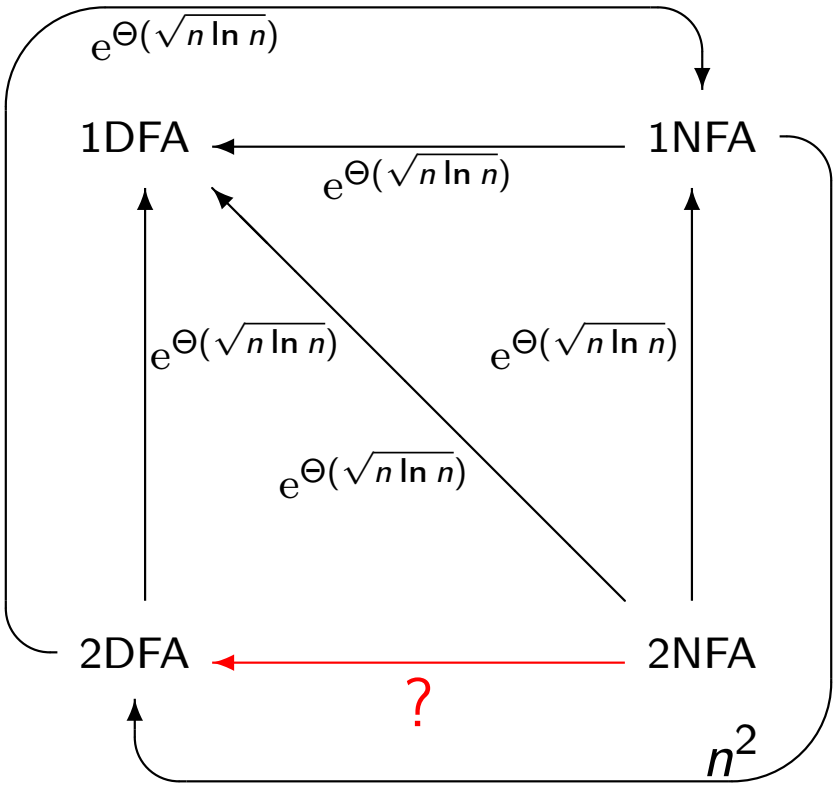



[Chrobak '86]

1NFA $\rightarrow$ 2DFA
Nel caso unario
questo problema è risolto!
(simulazione polinomiale)

# Simulazioni ottimali tra automi unari

I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale

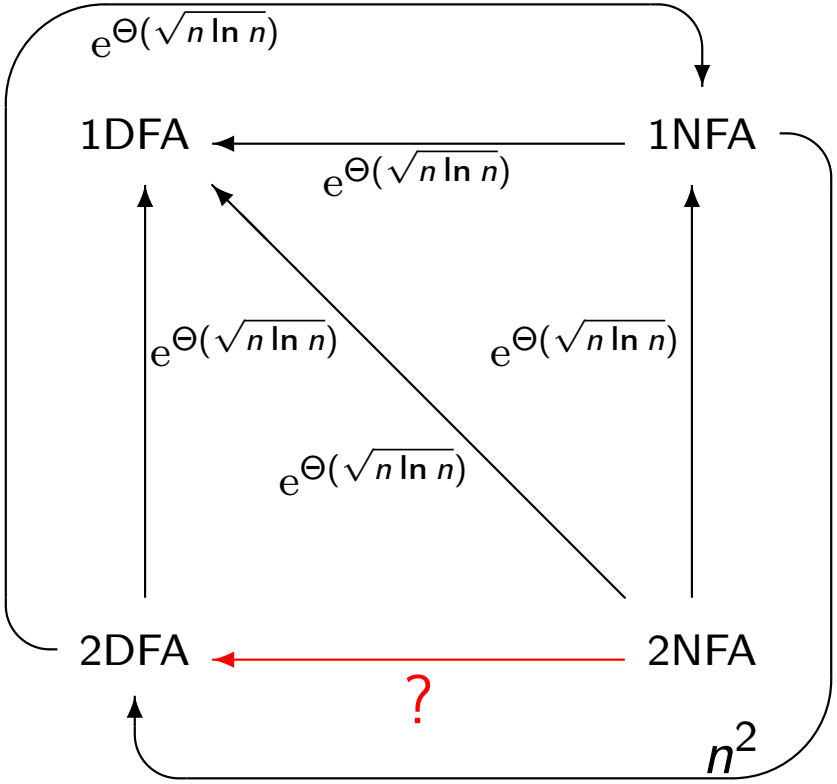



2NFA $\rightarrow$ 2DFA
Questo problema è aperto
*anche* nel caso unario!

- limite superiore $e^{\Theta(\sqrt{n \ln n})}$
  (da 2NFA $\rightarrow$ 1DFA)
- limite inferiore $\Omega(n^2)$
  (da 1NFA $\rightarrow$ 2DFA)

# Simulazioni ottimali tra automi unari

I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale

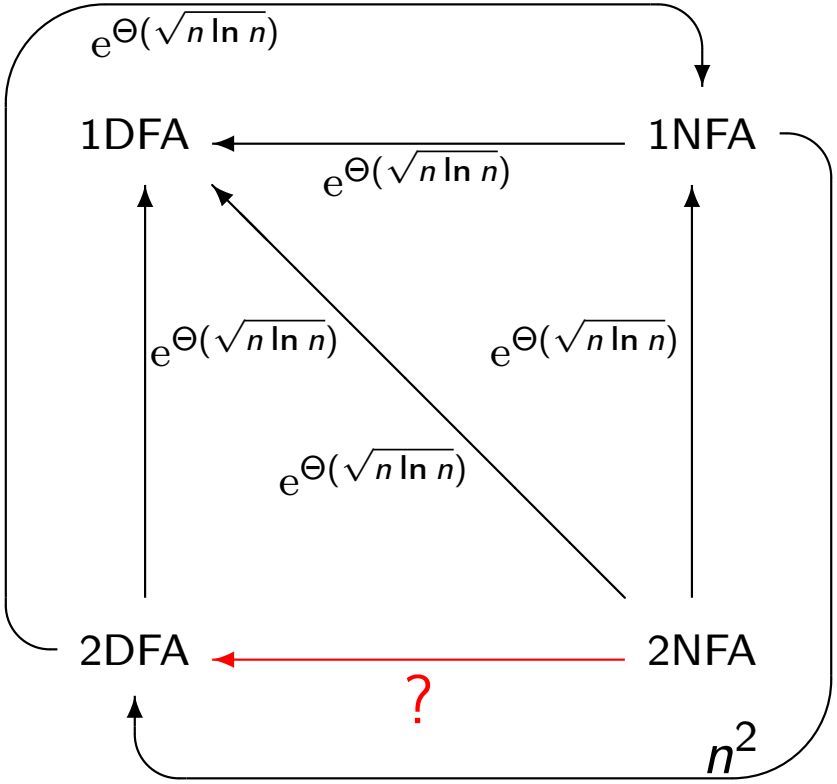



2NFA $\rightarrow$ 2DFA
Questo problema è aperto
*anche* nel caso unario!

- limite superiore $e^{\Theta(\sqrt{n \ln n})}$
  (da 2NFA $\rightarrow$ 1DFA)
- limite inferiore $\Omega(n^2)$
  (da 1NFA $\rightarrow$ 2DFA)

# Simulazioni ottimali tra automi unari

I costi delle simulazioni tra automi unari
sono differenti rispetto al caso generale




2NFA $\rightarrow$ 2DFA
Questo problema è aperto
*anche* nel caso unario!

- limite superiore $e^{\Theta(\sqrt{n \ln n})}$
  (da 2NFA $\rightarrow$ 1DFA)
- limite inferiore $\Omega(n^2)$
  (da 1NFA $\rightarrow$ 2DFA)

Il limite superiore è stato
ridotto a $e^{O(\ln^2 n)}$!

# Una forma normale per i 2NFA unari

[Geffert Mereghetti&P '03]

Automi *quasi sweeping* (qsNFA):

- *scelte nondeterministiche* e
- *inversioni di testina*

possibili *solo* quando la testina visita i delimitatori

**Teorema (Simulazione quasi sweeping)**

*Ogni 2NFA $A$ unario con $n$ stati può essere trasformato in un 2NFA $M$ t.c.*

- *$M$ è quasi sweeping*
- *$M$ ha al più $N \leq 2n+2$ stati*
- *$M$ e $A$ sono "quasi equivalenti"*
  *(possibili differenze solo su stringhe di lunghezza $\leq 5n^2$)*

# Una forma normale per i 2NFA unari

[Geffert Mereghetti&P '03]

Automi *quasi sweeping* (qsNFA):

- *scelte nondeterministiche* e
- *inversioni di testina*

possibili *solo* quando la testina visita i delimitatori

**Teorema (Simulazione quasi sweeping)**

*Ogni 2NFA $A$ unario con $n$ stati può essere trasformato in un 2NFA $M$ t.c.*

- *$M$ è quasi sweeping*
- *$M$ ha al più $N \leq 2n + 2$ stati*
- *$M$ e $A$ sono "quasi equivalenti"*
  *(possibili differenze solo su stringhe di lunghezza $\leq 5n^2$)*

# Simulazione quasi sweeping: conseguenze

(i) Simulazione subesponenziale di 2NFA unari mediante 2DFA
Ogni 2NFA unario con $n$ stati può essere simulato da un 2DFA con $e^{O(\ln^2 n)}$ stati
Tecnica *divide–et–impera* [Geffert:Mereghetti&P '03]

(ii) Complementazione polinomiale di 2NFA unari
Conteggio induttivo per qsNFA
[Geffert:Mereghetti&P '07]

(iii) Simulazione polinomiale di 2NFA unari mediante 2DFA *nell'ipotesi* L = NL
[Geffert&P '11]

(iv) "Disambiguazione" polinomiale di 2NFA unari
[Geffert&P '11]

# Simulazione quasi sweeping: conseguenze

(i) Simulazione subesponenziale di 2NFA unari mediante 2DFA

Ogni 2NFA unario con $n$ stati può essere simulato da un 2DFA con $e^{O(\ln^2 n)}$ stati

Tecnica *divide–et–impera* [Geffert Mereghetti&P '03]

(ii) Complementazione polinomiale di 2NFA unari

Conteggio induttivo per qsNFA

[Geffert Mereghetti&P '07]

(iii) Simulazione polinomiale di 2NFA unari mediante 2DFA *nell'ipotesi* L = NL

[Geffert&P '11]

(iv) "Disambiguazione" polinomiale di 2NFA unari

[Geffert&P '11]

# Simulazione quasi sweeping: conseguenze

(i) Simulazione subesponenziale di 2NFA unari mediante 2DFA

Ogni 2NFA unario con $n$ stati può essere simulato da un 2DFA con $e^{O(\ln^2 n)}$ stati

Tecnica *divide–et–impera* [Geffert Mereghetti&P '03]

(ii) Complementazione polinomiale di 2NFA unari

Conteggio induttivo per qsNFA

[Geffert Mereghetti&P '07]

(iii) Simulazione polinomiale di 2NFA unari mediante 2DFA *nell'ipotesi* L = NL

[Geffert&P '11]

(iv) "Disambiguazione" polinomiale di 2NFA unari

[Geffert&P '11]

# Simulazione quasi sweeping: conseguenze

(i) Simulazione subesponenziale di 2NFA unari mediante 2DFA

Ogni 2NFA unario con $n$ stati può essere simulato da un 2DFA con $e^{O(\ln^2 n)}$ stati

Tecnica *divide–et–impera* [Geffert Mereghetti&P '03]

(ii) Complementazione polinomiale di 2NFA unari

Conteggio induttivo per qsNFA

[Geffert Mereghetti&P '07]

(iii) Simulazione polinomiale di 2NFA unari mediante 2DFA *nell'ipotesi* L = NL

[Geffert&P '11]

(iv) "Disambiguazione" polinomiale di 2NFA unari

[Geffert&P '11]

# Simulazione quasi sweeping: conseguenze

(i) Simulazione subesponenziale di 2NFA unari mediante 2DFA

Ogni 2NFA unario con $n$ stati può essere simulato da un 2DFA con $e^{O(\ln^2 n)}$ stati

Tecnica *divide–et–impera* [Geffert Mereghetti&P '03]

(ii) Complementazione polinomiale di 2NFA unari

Conteggio induttivo per qsNFA

[Geffert Mereghetti&P '07]

(iii) Simulazione polinomiale di 2NFA unari mediante 2DFA *nell'ipotesi* $\mathrm{L} = \mathrm{NL}$

[Geffert&P '11]

(iv) "Disambiguazione" polinomiale di 2NFA unari

[Geffert&P '11]

# Restrizioni su 2NFA

Automi *outer nondeterministic* (OFA) [Guillon Geffert&P '12]:

- *scelte nondeterministiche* possibili
  solo quando la testina visita i *delimitatori*

# Restrizioni su 2NFA

Automi *outer nondeterministic* (OFA) [Guillon Geffert&P '12]:

- *scelte nondeterministiche* possibili
  solo quando la testina visita i *delimitatori*

Dunque:

- Nessuna restrizione sull'alfabeto
- Nessuna restrizione sulle inversioni
- *Transizioni deterministiche* sui "veri" simboli di input

# Automi outer nondeterministic (OFA)

I risultati ottenuti per il caso unaro
sono stati estesi a questi modelli 2OFA: [Guillon Geffert&P '12]

(i) Simulazione subesponenziale di 2OFA mediante 2DFA

(ii) Complementazione polinomiale di 2OFA

(iii) Simulazione polinomiale di 2OFA mediante 2DFA
*nell'ipotesi* L = NL

(iv) "Disambiguazione" polinomiale di 2OFA

Tecniche parzialmente differenti dal caso unario
Eliminazione dei loop

# Automi outer nondeterministic (OFA)

I risultati ottenuti per il caso unaro
sono stati estesi a questi modelli 2OFA: [Guillon Geffert&P '12]

(i) Simulazione subesponenziale di 2OFA mediante 2DFA

(ii) Complementazione polinomiale di 2OFA

(iii) Simulazione polinomiale di 2OFA mediante 2DFA
*nell'ipotesi* L = NL

(iv) "Disambiguazione" polinomiale di 2OFA

Tecniche parzialmente differenti dal caso unario
Eliminazione dei loop

# Automi outer nondeterministic (OFA)

I risultati ottenuti per il caso unaro
sono stati estesi a questi modelli 2OFA: [Guillon Geffert&P '12]

(i) Simulazione subesponenziale di 2OFA mediante 2DFA

(ii) Complementazione polinomiale di 2OFA

(iii) Simulazione polinomiale di 2OFA mediante 2DFA
*nell'ipotesi* L = NL

(iv) "Disambiguazione" polinomiale di 2OFA

Tecniche parzialmente differenti dal caso unario
Eliminazione dei loop

# Problema di Sakoda&Sipser: stato corrente

- Limiti superiori

| | 1NFA→2DFA | 2NFA→2DFA |
|---|---|---|
| caso unario e OFA | $O(n^2)$ ottimale | $e^{O(\ln^2 n)}$ |
| caso generale | esponenziale | esponenziale |

Caso unario [Chrobak '86, Geffert Mereghetti&P '03]
OFA [Guillon Geffert&P '12]

- Limiti inferiori

In tutti i casi, il miglior limite inferiore noto è $\Omega(n^2)$
[Chrobak '86]

# Osservazioni conclusive

Se si parla di...

**...automi a stati finiti**

di solito ci si riferisce a

*Automi one-way*

# Osservazioni conclusive

Se si parla di...

**...automi a stati finiti**

di solito ci si riferisce a
*Automi one-way*

**...Turing machines**

di solito ci si riferisce a
*Macchine di Turing two-way*

# Osservazioni conclusive

Se si parla di...

**...automi a stati finiti**

di solito ci si riferisce a
*Automi one-way*

**...Turing machines**

di solito ci si riferisce a
*Macchine di Turing two-way*

Perché questa differenza?

# Osservazioni conclusive

Se si parla di...

**...automi a stati finiti**

di solito ci si riferisce a
*Automi one-way*

**...Turing machines**

di solito ci si riferisce a
*Macchine di Turing two-way*

Perché questa differenza?

In entrambi i casi:

- *Computabilità*
- *Complessità*

# Osservazioni conclusive

Se si parla di...

**...automi a stati finiti**

di solito ci si riferisce a
*Automi one-way*

**...Turing machines**

di solito ci si riferisce a
*Macchine di Turing two-way*

Perché questa differenza?

In entrambi i casi:

- *Computabilità*
- *Complessità*

*Minicomplexity*

- Teoria della complessità per gli automi two-way

[Kapoutsis, DCFS 2012]

# Osservazioni conclusive

- Il problema di Sakoda e Sipser è una sfida molto stimolante
- Nello studio di casi particolari sono stati esaminati vari modelli interessanti e piuttosto naturali
- I risultati ottenuti in casi particolari, sebbene non risolvano il problema generale, sono non banali e, in vari casi, molto profondi
- Legami con la complessità strutturale e in spazio
  - problemi
  - tecniche
- Legami con la teoria dei numeri (automi unari)

# Osservazioni conclusive

- Il problema di Sakoda e Sipser è una sfida molto stimolante
- Nello studio di casi particolari sono stati esaminati vari modelli interessanti e piuttosto naturali
- I risultati ottenuti in casi particolari, sebbene non risolvano il problema generale, sono non banali e, in vari casi, molto profondi
- Legami con la complessità strutturale e in spazio
  - problemi
  - tecniche
- Legami con la teoria dei numeri (automi unari)

# Osservazioni conclusive

- Il problema di Sakoda e Sipser è una sfida molto stimolante
- Nello studio di casi particolari sono stati esaminati vari modelli interessanti e piuttosto naturali
- I risultati ottenuti in casi particolari, sebbene non risolvano il problema generale, sono non banali e, in vari casi, molto profondi
- Legami con la complessità strutturale e in spazio
  - problemi
  - tecniche
- Legami con la teoria dei numeri (automi unari)

# Osservazioni conclusive

- Il problema di Sakoda e Sipser è una sfida molto stimolante
- Nello studio di casi particolari sono stati esaminati vari modelli interessanti e piuttosto naturali
- I risultati ottenuti in casi particolari, sebbene non risolvano il problema generale, sono non banali e, in vari casi, molto profondi
- Legami con la complessità strutturale e in spazio
  - problemi
  - tecniche
- Legami con la teoria dei numeri (automi unari)

# Osservazioni conclusive

- Il problema di Sakoda e Sipser è una sfida molto stimolante
- Nello studio di casi particolari sono stati esaminati vari modelli interessanti e piuttosto naturali
- I risultati ottenuti in casi particolari, sebbene non risolvano il problema generale, sono non banali e, in vari casi, molto profondi
- Legami con la complessità strutturale e in spazio
  - problemi
  - tecniche
- Legami con la teoria dei numeri (automi unari)

Grazie per l'attenzione!